Getting Started with Vernier Go Direct® Sensors and Web VPython

v230308

This guide shows you how to get started writing Web VPython programs to connect to and gather data from most Vernier Go Direct devices. The guide contains the following topics:

Notes:
This guide is for using web-based Web VPython. We have a separate guide for using Go Direct devices with installed VPython on your computer. See Getting Started with Vernier Go Direct Sensors and VPython at https://www.vernier.com/engineering/python/ . Look for the section Python and Go Direct Sensors.

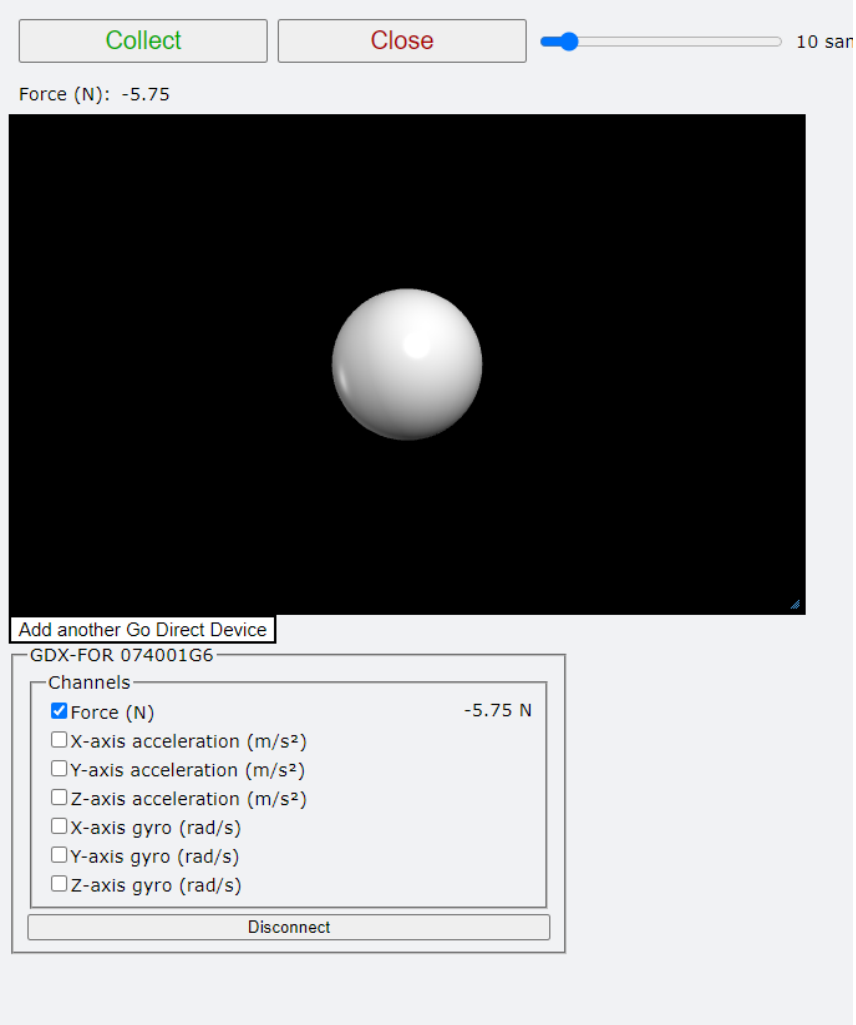
Most Vernier Go Direct devices work great using Web VPython. Some of the most complex and expensive Go Direct devices are not easy to use with do-it-yourself programs. Go Direct spectrometers, Mini GC, Polarimeter, Go Wireless Heart Rate, and Cyclic Voltammetry System do not work with Web VPython. Other Go Direct devices that are not supported, or that may require advanced programming, calibration, or analysis, include Blood Pressure, Ion-Selective Electrode, Optical Dissolved Oxygen, and some timing/event-based devices like Photogates, Drop Counters, and Projectile Launchers.

Overview of Web VPython

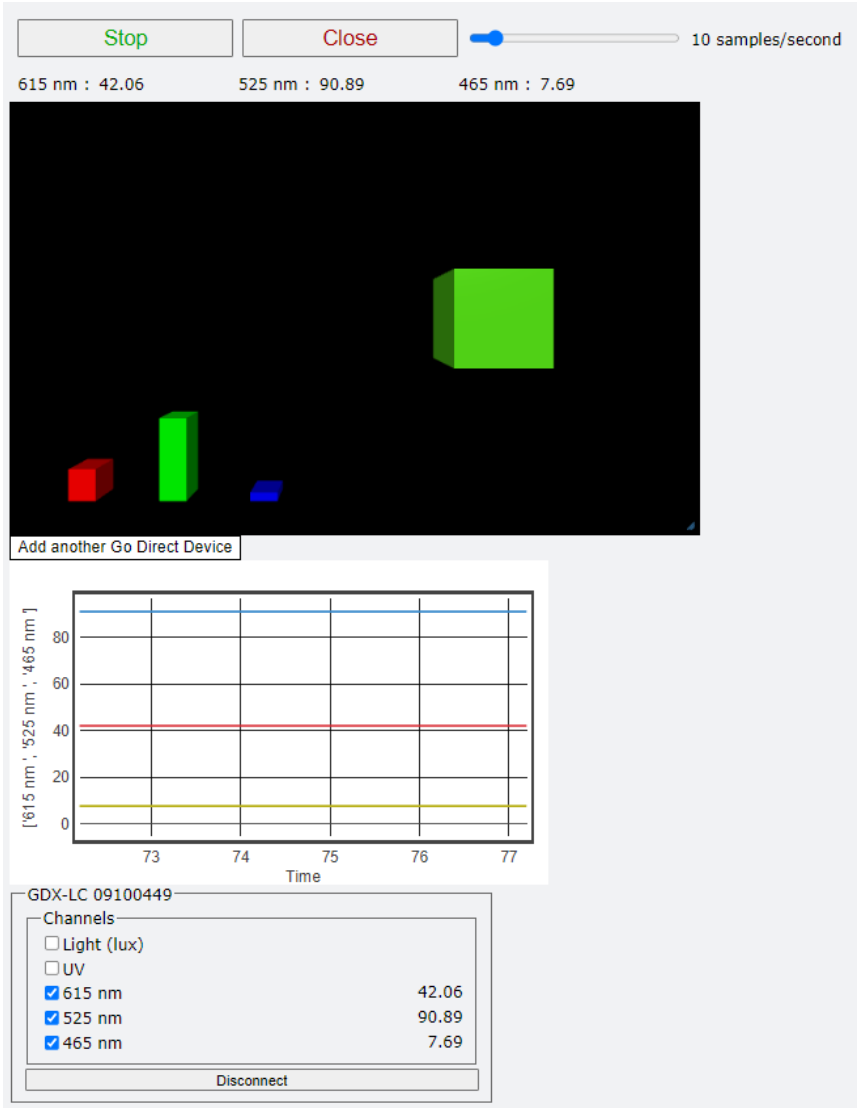
Web VPython (formerly known as Glowscript) is an easily accessible, browser-based coding platform that bundles a Python editor and the VPython library together, allowing users to easily create 3D graphics. It was developed to simplify the creation of physics animations and simulations in Python. Web VPython runs on Windows and Mac computers, or Chromebooks. Because it runs in a browser, there is nothing to install, although 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 (“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 .

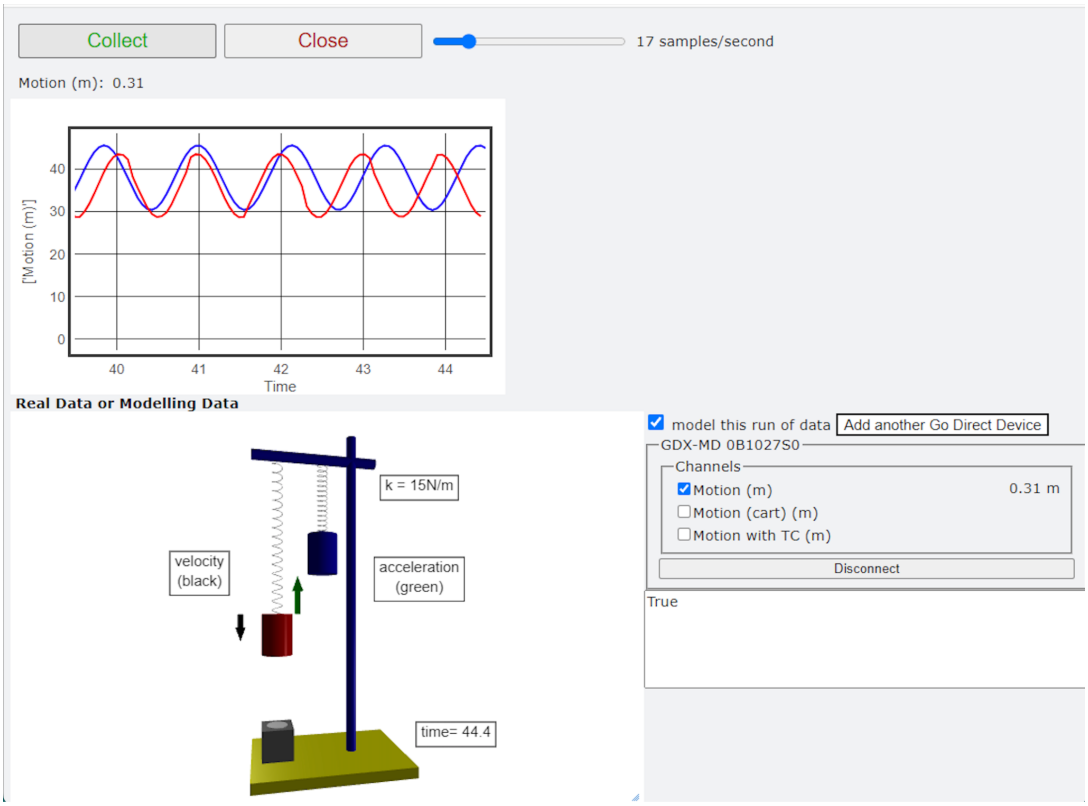
Here are some sample screenshots from Web VPython using Go Direct sensors:



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



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



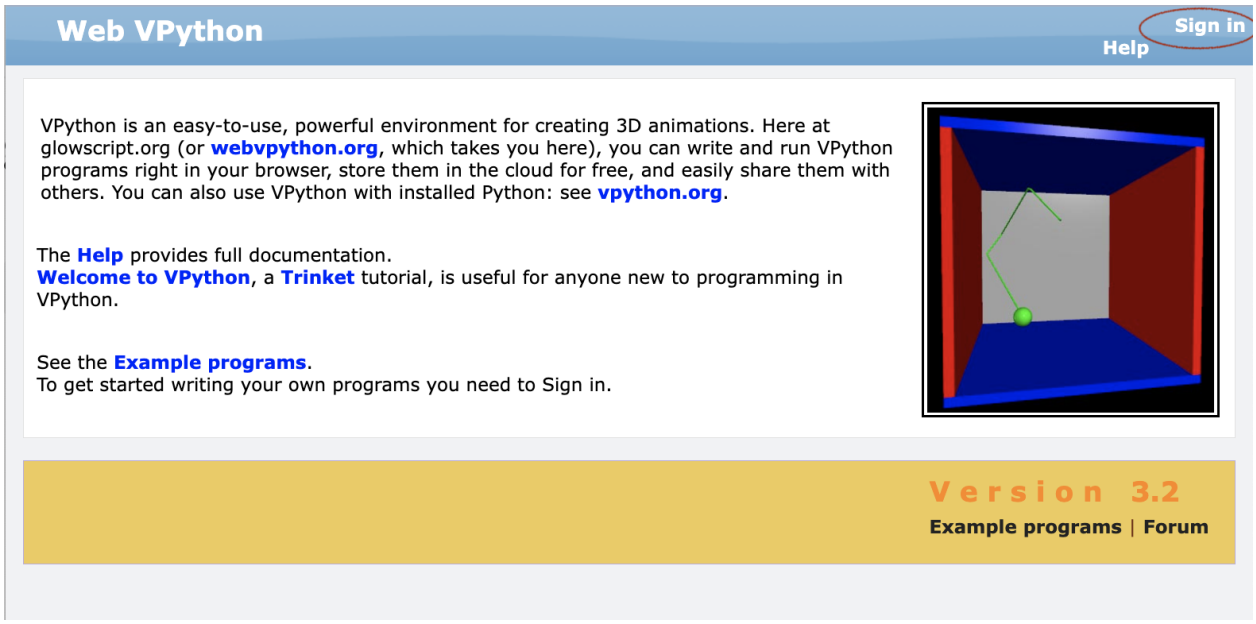
Simple harmonic motion program that compares collected data from Go Direct Motion Detector (GDX-MD) and a theoretical model

Getting Started Requirements

- A Vernier Go Direct Device
- A Windows® 10 or macOS® computer or a Chromebook
- Access to webvpython.org using the Chrome browser (The Chrome or Edge browser is the recommended browser, other browsers may not work)
- A 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*

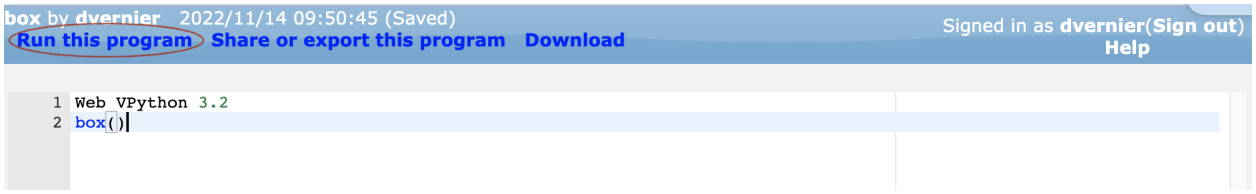
Your work in Web VPython is automatically saved to your account online. Your account includes rudimentary file management tools to organize your programs in folders, as well as rename, copy, and delete programs.

We would recommend clicking on the Example programs link and looking at the kind of things that can be done in Web VPython.

Create a Simple Web VPython Program

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

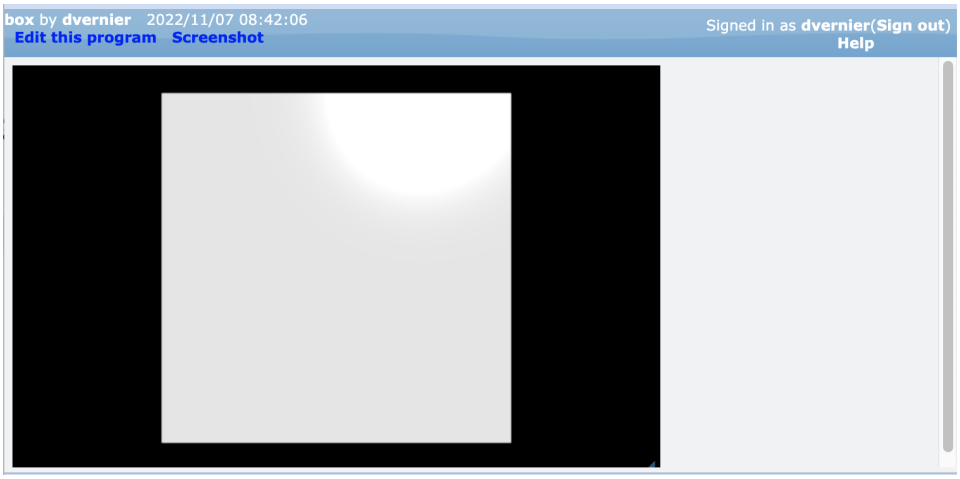
1. Go to www.webvpython.org
2. Sign in to your account.
3. Go to your programs (click on your sign in name, or click the link that takes you to your programs).
4. Once inside your account, click **Create New Program**.
5. Name your new program “box” and click **Create**.
6. In line 2, write the following 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.
- 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.

Web VPython Program with a Go Direct Device

It is easy to incorporate Go Direct sensor data into your Web VPython program! Take a look at the following example program, called “VPythonGettingStartedUSB”. This example can be found at

https://www.glowscript.org/#/user/vernier-web-vpython/folder/MyPrograms/program/VPythonGettingStartedUSB

Let's go through this program line-by-line to explain what each line is doing:

Web VPython 3.2
The first line is required for all Web VPython programs.

get_library("https://unpkg.com/@vernier/godirect/dist/webVPython.js")
This line provides the link to JavaScript libraries that allow Web VPython to communicate with our Go Direct devices. You must have internet connectivity to use these libraries.

gdx.open(connection='usb')
This function tells the program how you would like to connect to your Go Direct device. Change the 'usb' to 'ble', if you want to connect using Bluetooth.

gdx.select_sensors()
This function specifies which of the channels of your Go Direct device you want to read. When the argument is left blank, the default channel for the sensor is used. You can also select the channel from the Web VPython canvas while the program is running.

gdx.vp_vernier_canvas()

Use this function to add VPython objects that are useful for data collection to the VPython canvas. When the argument is left blank a Collect/Stop button, Close button, a slider for controlling data collection rate, a live meter readout, and a channel setup box will be added to the canvas. A graph object may be added by including `graph=True` in the argument.

`gdx.start()`
This function sets the data collection period in milliseconds. If the argument is left blank it will set the data collection period to 100 ms. You can set the period with an argument (`period=100`). Once the canvas opens, you can modify the sample period by using the slider on the VPython canvas while the program is running.

```
ball = sphere(pos=vector(0,1,0,radius=1,color=color.red)
```
This is standard VPython code that creates a 3D ball object. It is easy to change the ball's position, size, color, or other properties.

```
while gdx.vp_close_is_pressed() == False:
        rate(500)
        while gdx.vp_collect_is_pressed() == True:
                rate(500)
```
This is a nested data-collection loop. The outer loop monitors the Close button. If the Close button is pressed, both loops are terminated and the program ends. As long as the Close button has not been pressed the inner loop will monitor 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() statements are VPython statements that are required in an animation loop. They prevent the program from monopolizing the browser when nothing is happening in the loop.

```
measurements = gdx.read()
```
This line reads a data point from all of the sensor channels that you have selected and stores them in a variable called 'measurements'. The function gdx.read() returns data as a 1-dimensional list. If only 1 channel is active, this will still be a list, just a list with one value.

```
if measurements == None:
    Break
```
These lines handle the situation when there is no data to read (which should only occur if there is an error with the Go Direct device). This will terminate the data collection loop.

```
ball.radius=measurements[0]
```
As stated above, the 'measurements' variable is a list. The first value of a list has an index value of 0. Therefore, measurements[0] gives the sensor value and sets the radius of the ball as this value.

More detailed information for all of the functions can be found below.

Run Web VPython with a Go Direct Device

Give the program above (“VPythonGettingStartedUSB”) a try. Sign into your account, go to your programs and click on **Create New Program**. Notice that you get the start of a new Web VPython program (the first line only). There are several options on how to create the program:

Type the program in yourself.

or:
(if you are reading this electronically) Copy the text below and paste it in. This is just the whole program with the first line left out.

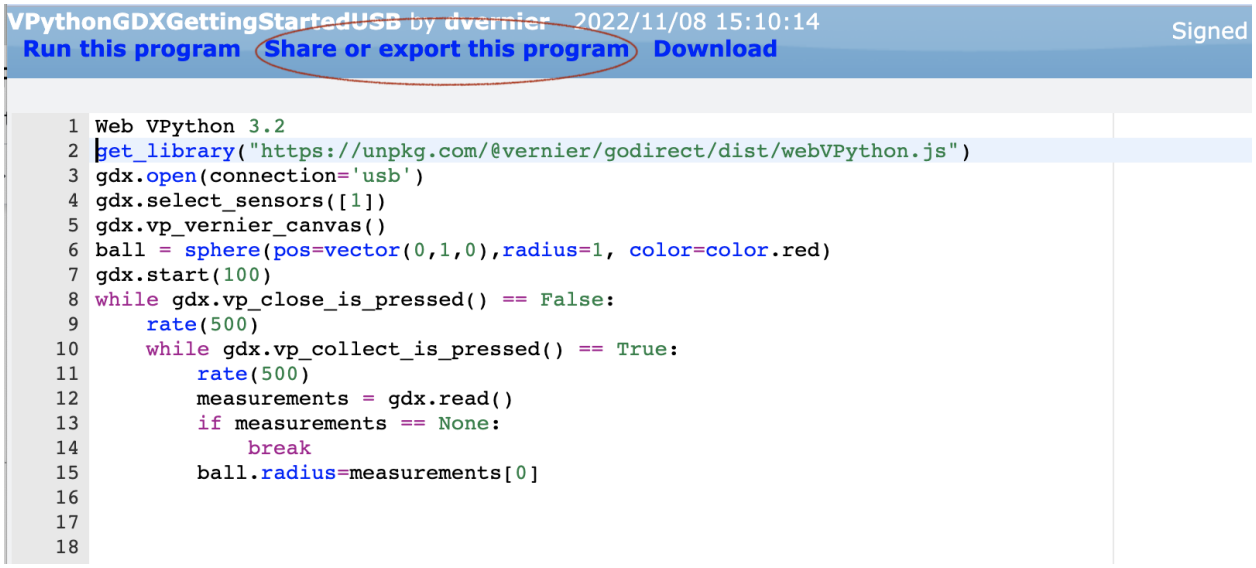
```
get_library('https://unpkg.com/@vernier/godirect/dist/webVPython.js')
gdx.open(connection='usb')
gdx.select_sensors()
gdx.vp_vernier_canvas()
gdx.start()
ball=sphere(pos=vector(0,1,0),radius=1,color=color.red)
while gdx.vp_close_is_pressed() == False:
      rate(500)
      while gdx.vp_collect_is_pressed() == True:
            rate(500)
            measurements = gdx.read()
            if measurements == None:
                  break
            ball.radius=measurements[0]
```

Or:
Go to our web site at https://www.glowscript.org/#/user/vernier-web-vpython/folder/MyPrograms/

Find the VPythonGetttngStartedUSB program and click on **View** to see the program. Copy all the text and paste it into your new program. Be careful not to duplicate the first line.

Whatever method you use, when you have the program typed in, you will see "Run this program" in the editor. That option will work for VPython code that does not include Go Direct devices, but is not the way to run a Web VPython program that includes a Go Direct device. For internet security reasons, you must download your program as an HTML file and then later run it. Here is the process:

Choose **Share or export this program**.



A new web page will open. Note the whitespace at the bottom of this page. If this area is blank there is a bug in the code. Go back to the editor to fix the code. If there is code in the white space, click on the **Download as HTML** link

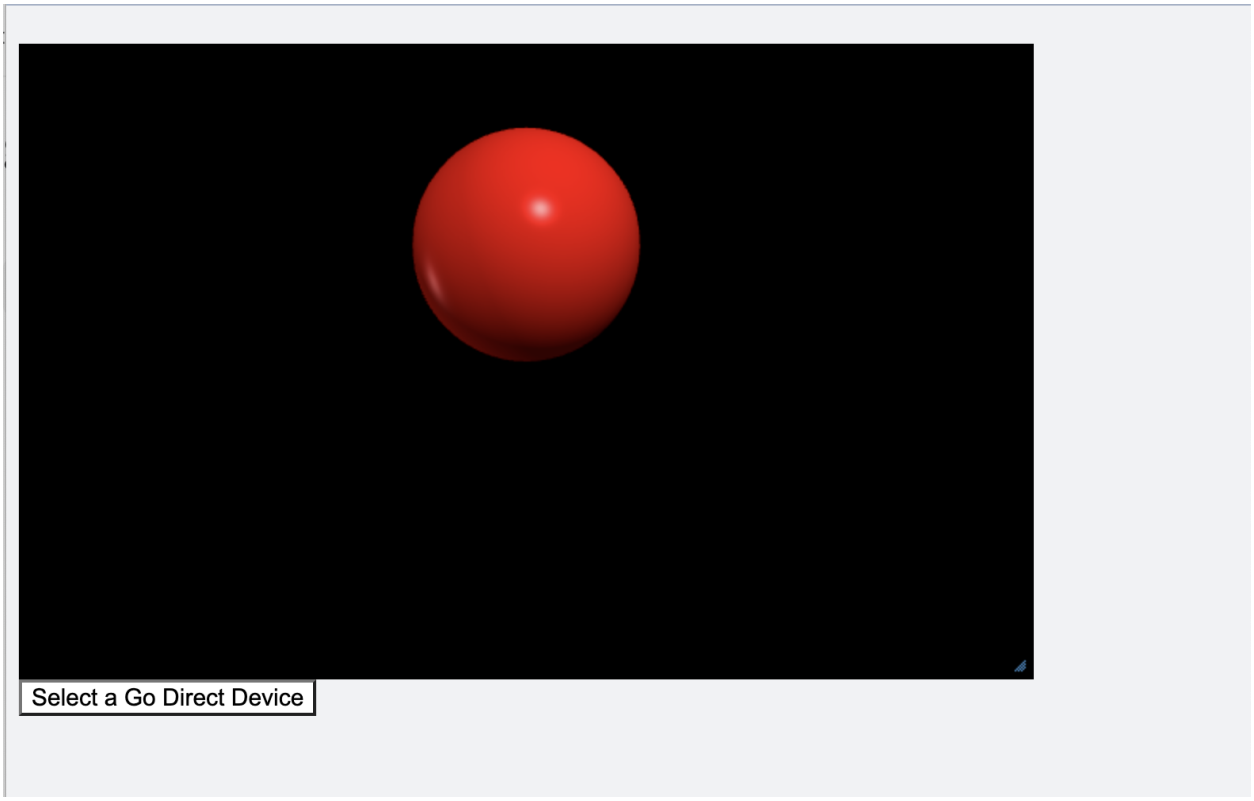
Once the html file has downloaded to your computer, click on the html file. This will open and run the program in your browser. If you are using a Chrome browser, an icon for the downloaded file will usually appear at the bottom left of the browser and you can just click on that to run it. The file created will actually be in your Downloads folder so you can also go there and double-click on it to run it.

Notes about this process:

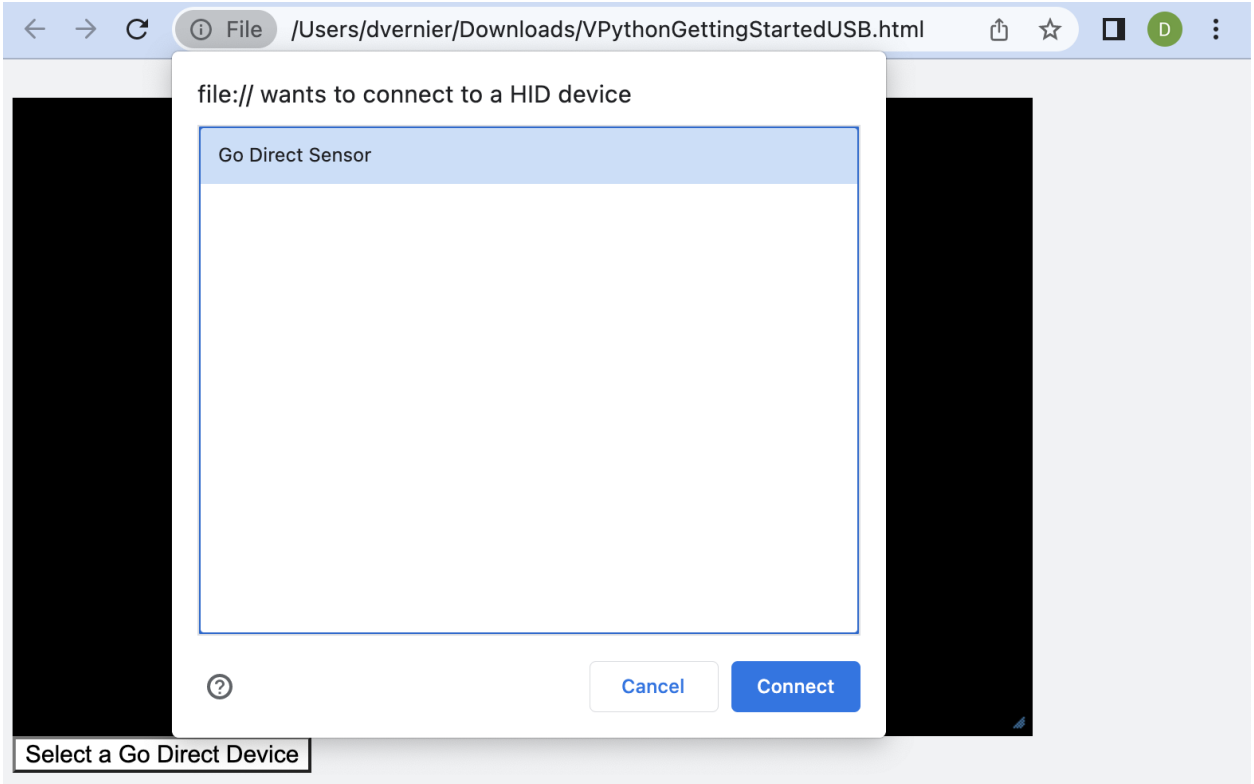
- Now, you will probably think this is a clumsy way to run a program, and it is, but it works well and you get used to it. One advantage of this, is that you are creating an executable HTML file that can be used by anyone, even if they have never heard of Python or Web VPython. You can even send the file in an email and have someone else run it.

- Another downside of this system is that you end up with lots of downloads and lots of open tabs in your browser. You will need to do periodic cleanup operations.

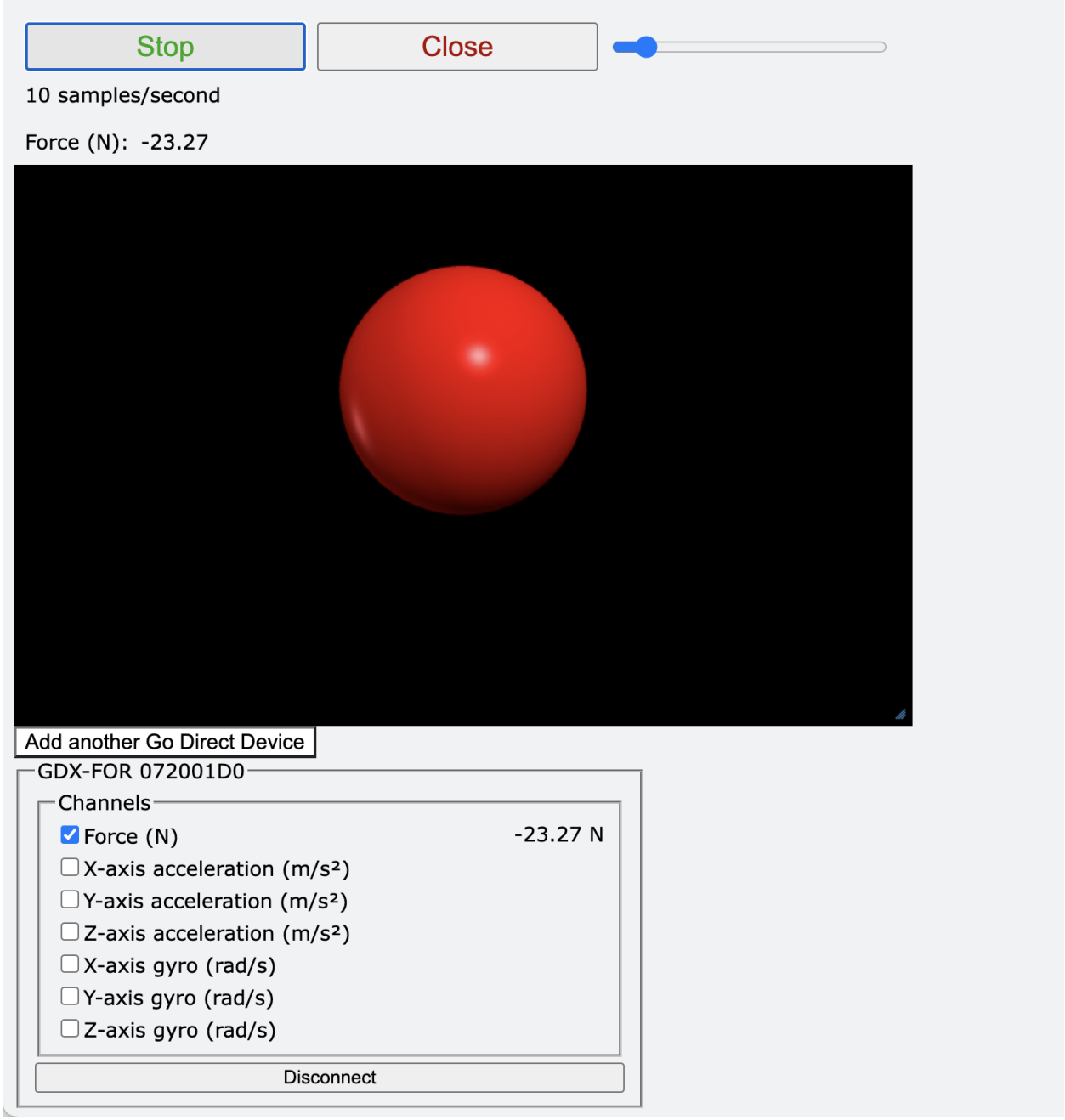
When the program runs, click on the "**Select a Go Direct Device**" button.



A dialog box will open with a list of Go Direct devices. Click on the name of the device to highlight it and then click "Connect". Often only one device will be listed.



When the 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.



Click on the "**Collect**" button to start data collection. Your sensor reading will control the size of the box.

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 computer.

Example Programs

Find our examples for Web VPython using Go Direct devices at:

https://www.glowscript.org/#/user/vernier-web-vpython/folder/MyPrograms/

Here is a list of some of the programs available.

VPythonGettingStartedUSB - The program discussed above. Use it with almost any GDX sensor

Gdxlc-color-match - use with a GDX-LC Light and Color Sensor

ForceVectors - use with a GDX-FOR Force Sensor

Gdxacc-control-tilt - use with a GDX-ACC Acceleration Sensor

Gdxhd-control-tilt - use with a GDX-HD Hand Dynamometer Sensor

G**dxfor-live-freebody-diagram** - Use with a GDX-FOR Force Sensor

Gdxmd-simple-harmonic-oscillator - Use with a GDX-MD Motion Sensor. This is a complex program which can be used to compare measured position data with model data.

MappingElectricPotentialDemonstration - No sensor needed. This is a demonstration of plotting 2-D and 3-D field maps in Web VPython. Since no sensor is connected, you can just Run this program without having to download it as an HTML file.

The programs below are simple programs used to introduce Web VPython and its use with GDX Sensors.

ex1-vpython-object

ex2-vpython-modify-attributes

ex3-vpython-box-length

ex4-vpython-sphere-pos

ex5-vpython-gdx-graph

The Web VPython Functions

Here is some more information about the gdx functions, including how you might add arguments to a few of the functions:

gdx.open(USB) or gdx.open(connection=USB)

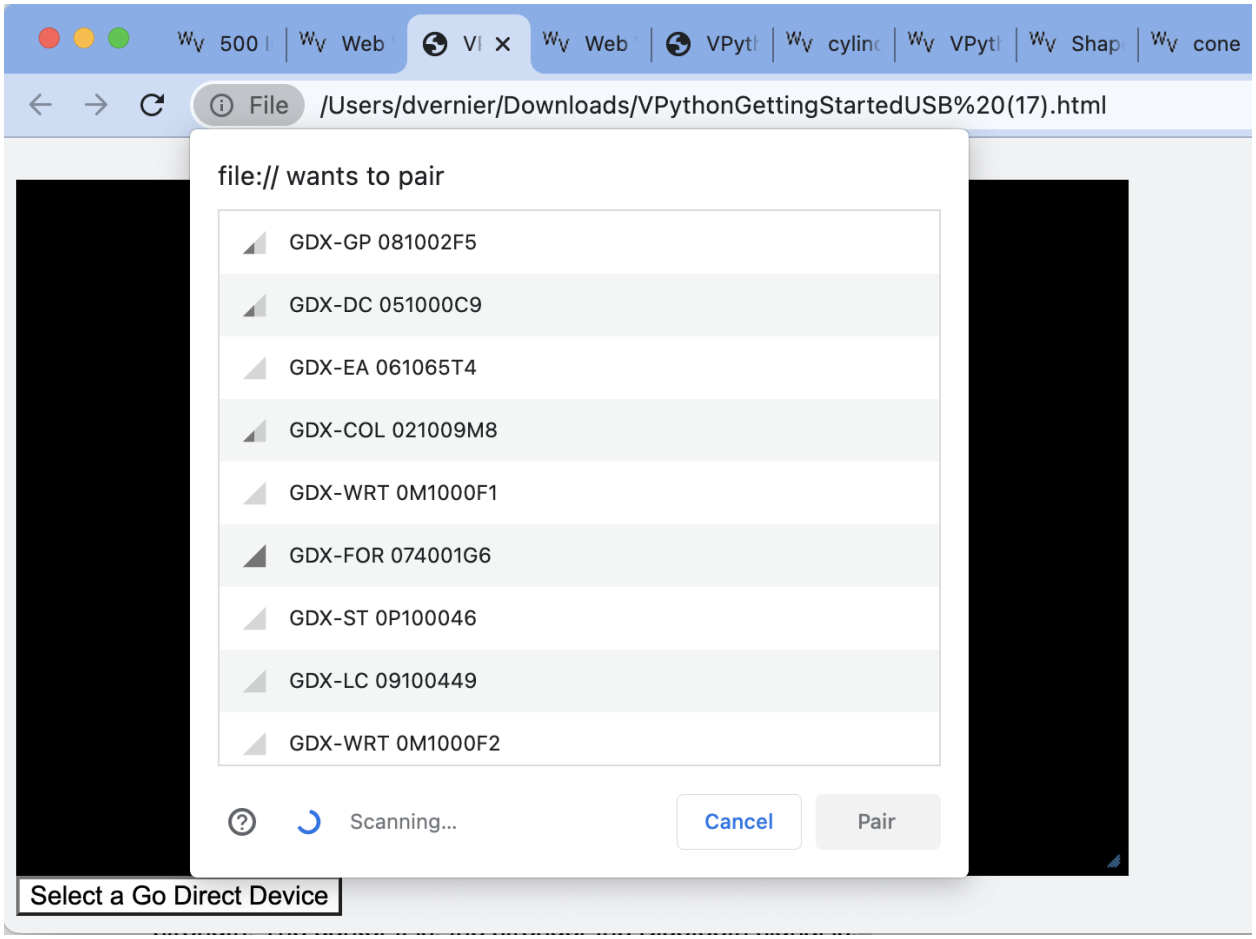
- The connection= parameter can be set to connect Go Direct devices by USB 'usb' or Bluetooth 'ble'

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

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

Also, you can shorten the command to gdx.open(‘usb’) or gdx.open(‘ble’). You can also use the upper case letters “USB” and “BLE”.

Note that if you are using a Bluetooth connection, when you see the list of available sensors, there is a wedge-shaped icon displayed. This is an indication of the Bluetooth radio signal strength. The darker it is, the stronger the Bluetooth signal is. Here is an example made in a room with lots of GDX devices turned on.



gdx.select_sensors()

Most of the time, you can just use `gdx.select_sensors()`, with nothing inside the parentheses. This will open the default channel on the Go Direct device. If you want to specify the channel or channels on the sensor to be read, note that the format is a list of sensor channels inside square brackets like this

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

Or, if you want to use multiple channels on one GDX device, separate the channel numbers with commas::

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

Note that the channels available on most GDX devices are listed later in this manual. You can use channel 1 for almost any sensor. Two exceptions are: GDX-MD and GDX-RMS. The default is channel 5 for them. Also, do not use channel 1 with GDX-SND (because it needs to sample very quickly on that channel). Unfortunately, channel 1 is the default channel for GDX-SND, so that is one GDX sensor that will not work properly in Web VPython if you use gdx.select_sensors().

Note that the user can leave the argument blank and then specify the sensor channels by clicking in the Channel Setup Box while the Web VPython program is running.

gdx.start()

This command sets the sampling rate, that is, the time between samples in milliseconds. The same rate is used for all selected sensors.

If this function's argument is left blank, a default sampling period of 100 milliseconds (10 readings per second) will be chosen. The user can always modify this using the slider on the canvas.

Sampling at a period that is less than 10 milliseconds (in other words, greater than 100 samples/second) may be problematic.

gdx.read()

The `gdx.read()` function will take single point readings from the selected sensors at the desired period and return the readings as a one dimensional list. The first item in the list is measurements[0], the second is measurements[1], etc.

Place this function in the data collection loop and make sure the loop can iterate fast enough to keep up with the sampling period. You do not need to add other code in the loop to slow down the loop.

gdx.vp_close_is_pressed()

The function `gdx.vp_close_is_pressed()` monitors the state of the VPython canvas CLOSE button.

gdx.vp_collect_is_pressed()

The function `gdx.vp_collect_is_pressed()` monitors the state of the VPython canvas COLLECT/STOP button.

gdx.vp_get_slider_period()

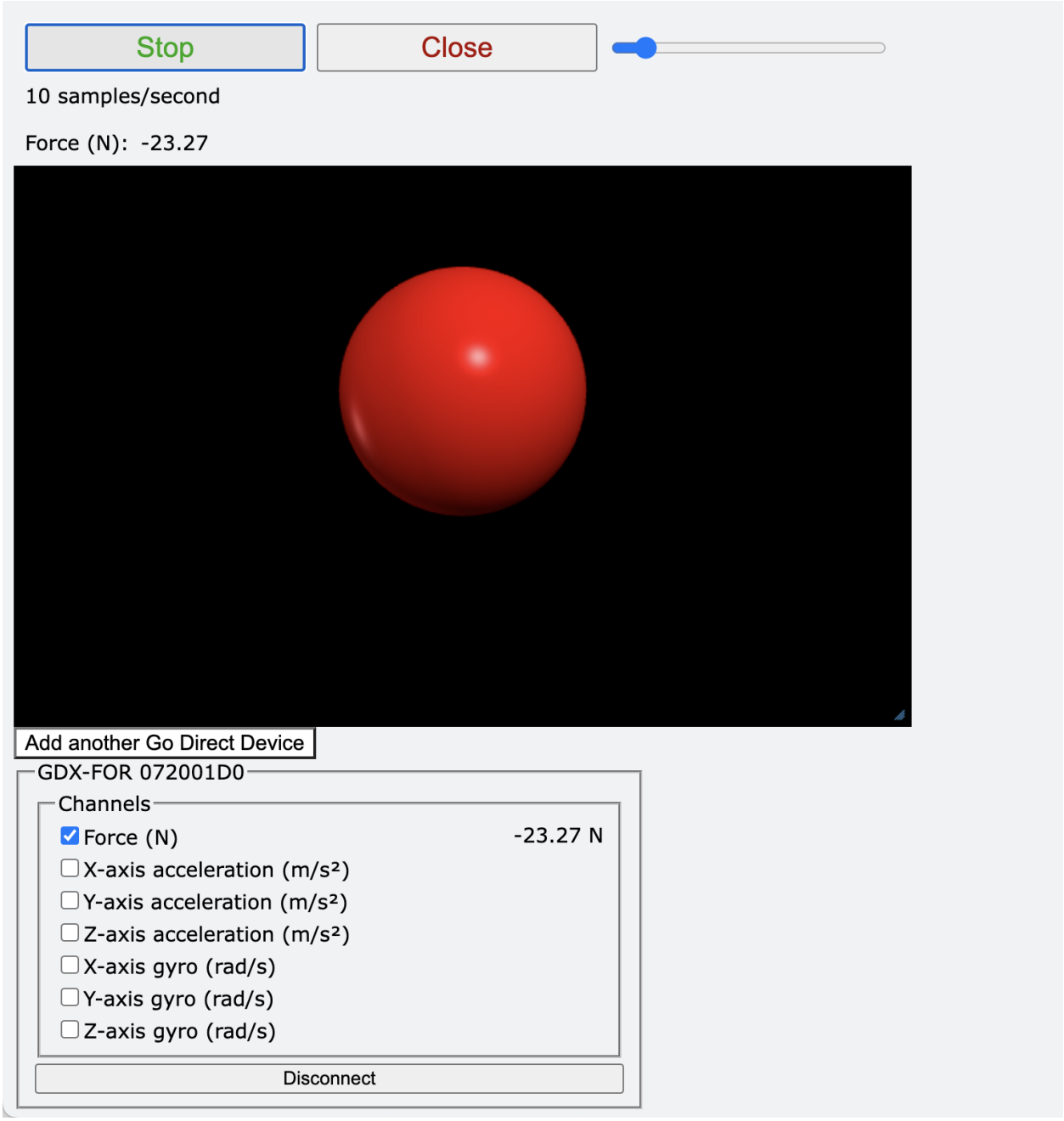
This is not used in our simple sample program above. It returns the period of data collection in milliseconds. It is sometimes needed because the slider can change the data collection rate and other parts of the program may need to know about that change.

About the Vernier Canvas

The function gdx.vp_vernier_canvas() adds some features that are often nice for programs involving data collection. The default form:

```
gdx.vp_vernier_canvas()
```

will set up a display like this:



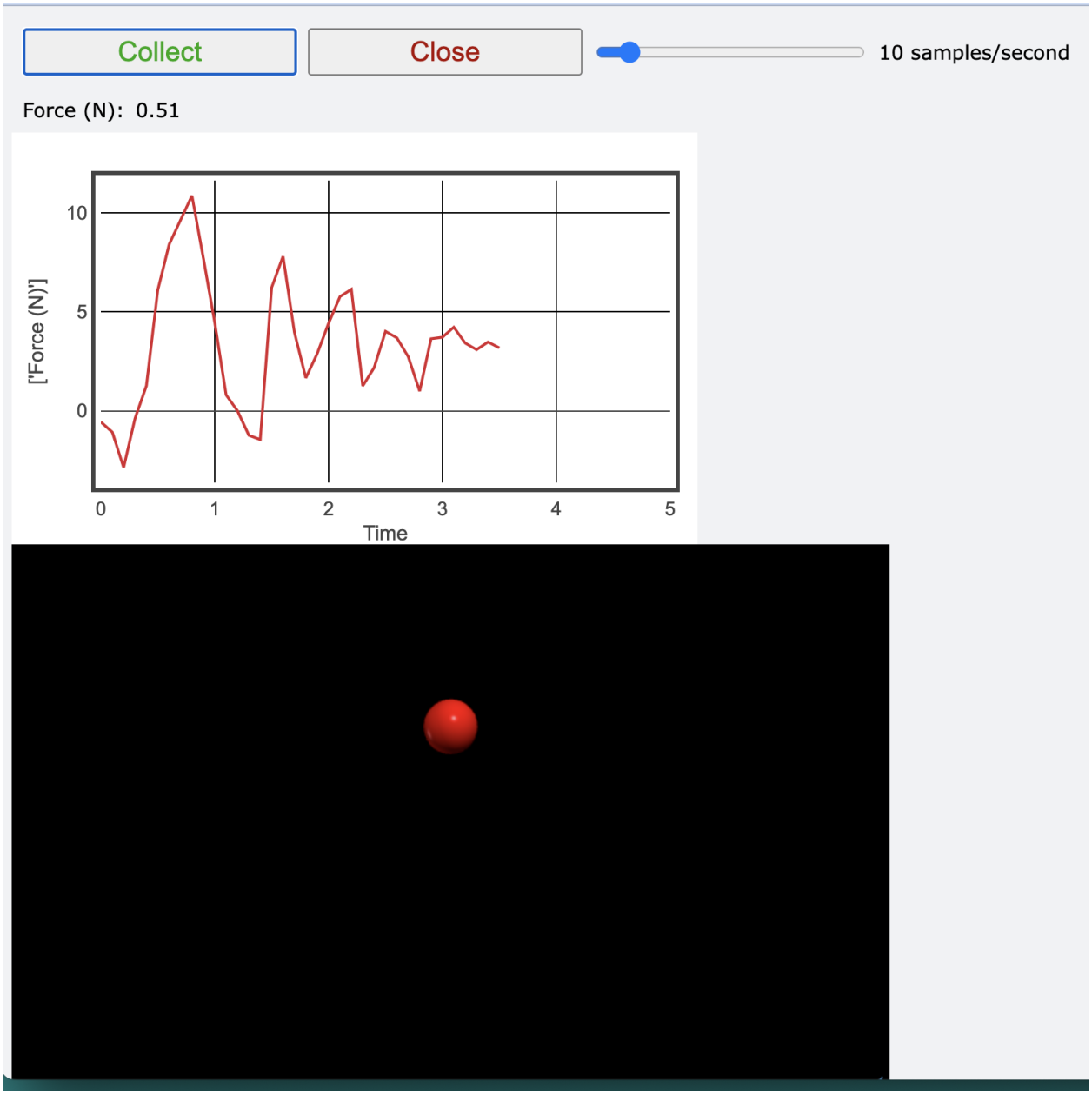
Any GDX data collection program can then be set up so that the COLLECT button starts data collection. It will change to STOP and then you can stop data collection. These operations can be repeated as you like. The CLOSE button will nicely shut down the GDX connection with the computer.

Notice on the sample screen above there is a slider to the right of the buttons that can be used to control the data collection rate of the GDX sensor.

After you are connected to a sensor, by default, below the main canvas, you will see a Channel Setup box. This shows the channels that are active on the connected Go Direct device. If you

do not want the channel setup box, you can eliminate it by modifying your `gdx.vp_vernier_canvas()` statement to:
`gdx.vp_vernier_canvas(channel_setup=False)`

You can also add a graph by adding "`chart=True`" to the gdx.vp_vernier_canvas() function. The function `gdx.vp_vernier_canvas(channel_setup=False, chart=True)` used with the simple program we have been using yields:



Note that by default, a graph is not included on the Vernier canvas. You have to specify that you want it. The complete format for the `gdx.vp_vernier_canvas` function is:

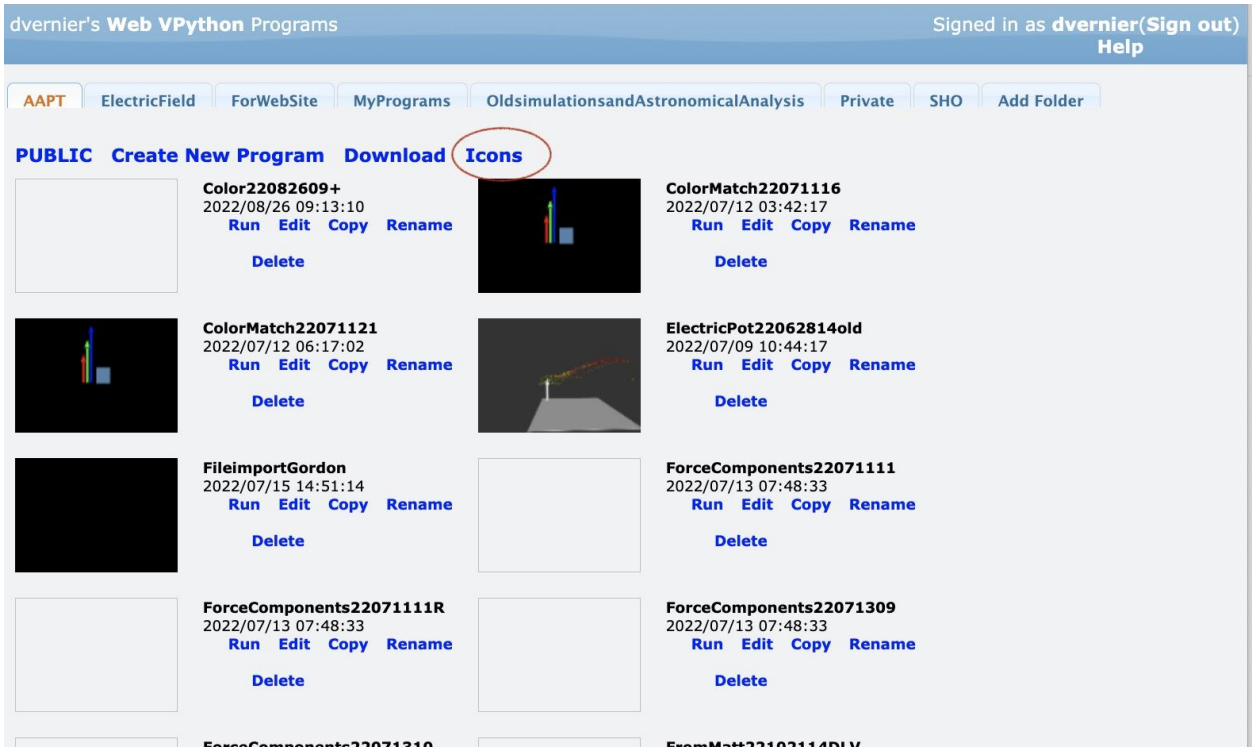
```
gdx.vp_vernier_canvas(buttons=True, meters=True, slider=True,
channel_setup=True, chart=False)
```

The statement above has the default settings. In short, you will get the buttons, slider, meters, and the channel setup box by default, and you have to specifically turn on the graph feature, if you want it.

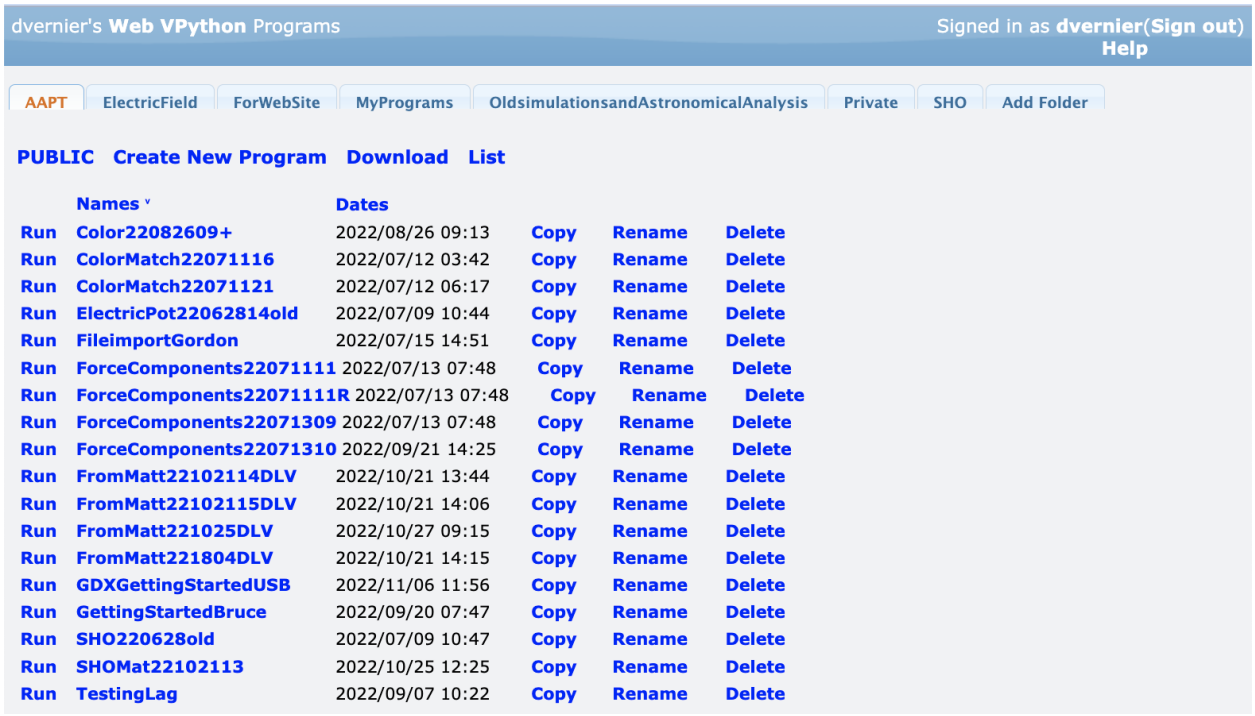
Tips for using Web VPython

Displaying Programs as Lists

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.



Like this:



The primary advantage of the List view is that you can sort the programs by name or date, by clicking on the column heading. You can also create a folder and rename or delete programs.

Moving between Web VPython and Installed VPython

As mentioned earlier, you can also use Go Direct devices with installed VPython on your computer; for details, see Getting Started with Vernier Go Direct Sensors and VPython. Occasionally you will 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 this:

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

2. Add these three lines:

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

If you want to move an installed Python program into a format that can be run in Web VPython, do the following:

1. Add these two lines to the top of your Web VPython 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:
    - 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 two rate() statements in the Web VPython data collection loop. The `rate()` statements do not appear to be as important when working within installed VPython.

Channels Available on Go Direct Devices

For most Go Direct devices, you can just use channel 1. Even better, usually you can just use the
`gdx.select_sensors()`

With nothing in between the parentheses, the program will use the default channel. This includes temperature, force, light, acceleration, voltage, current, energy, magnetic field, CO2, O2, colorimeters, conductivity, and carts. Here is a list of available channels on some more complex Go Direct devices.

Below is a list of the Go Direct devices most commonly used with Web VPython and the channels they have available. There are also notes about unusual things about the channel options.

GDX-3MG
1 X magnetic field
2 Y magnetic field
3 Z magnetic field
4 X magnetic field 130mT
5 Y magnetic field 130mT
6 Z magnetic field 130mT

GDX- ACC
1 through 3 acceleration X,Y,Z (m/s^2)
4 through 6 acceleration X,Y,Z (high range)(m/s^2)
7 through 9 gyro (radians/s)
10 altitude(m)
11 angle (degrees)

GDX-CART
1 Position
2 Force
3 X Acceleration
4 Y Acceleration
5 Z Acceleration

GDX-CCS
1 Current

GDX-CO2
1 CO2 Gas
2 Temperature
3 Relative Humidity

GDX-COL
1 Transmittance

GDX-CON
1 Conductivity 0% Temperature Compensated
2 Conductivity
3 Temperature

GDX-DC
1 Volume

GDX-EA (used for pH)
1 Potential
2 pH

GDX-EKG
1 EKG
2 Heart Rate
3 EMG
4 EMG Rectified
5 Voltage

GDX-ETOH
1 Ethanol Vapor

GDX-ISEA
1 Potential
2 Nitrate
3 Ammonium
4 Calcium
5 Chloride
6 Potassium

GDX-HD
1 Force
2 X-axis acceleration
3 Y-axis acceleration
4 Z-axis acceleration
5 X-axis gyro
6 Y-axis gyro
7 Z-axis gyro

GDX-NRG
1 Potential
2 Current
3 Power
4 Resistance
4 Energy

GDX-LC
1 Light (lux)
2 UV
5 615 nm
6 525 nm
7 465 nm

GDX-MD (note that this sensor is unusual in that channel 5 is the default channel)
5 Motion
6 Motion (cart)
7 Motion with TC
(all incompatible with one another)

GDX-O2
1 O2 Gas
2 O2 Gas Temperature Compensated

3 Temperature

GDX-ODO
1 DO Concentration
2 DO Saturation
3 Temperature
4 Pressure
5 DO Salinity

GDX-Q
1 Charge
2 Potential

GDX-RB
1 Force
2 Respiration Rate
4 Steps
5 Step Rate

GDX-RMS (note that this sensor is unusual in that channel 5 is the default channel)
5 Angle
6 Angle (high resolution)

GDX-SND (note that this sensor is unusual and problematic in that the default channel takes data at a very high rate and is not easy to read in Web VPython)
1 Sound Pressure (needs to be sampled at a high speed)
2 Sound Level (A weighted) (can be as slow as sampled a few times a second)
3 Sound Level (C weighted) (can be as slow as sampled a few times a second)
4 Wave Amplitude

GDX-ST
1 Temperature

GDX-TMP
1 Temperature

GDX-WRT
1 Temperature

GDX-WTHR
1 wind speed (m/s)
2 wind direction (degrees)
3 wind chill (degrees C)
4 temperature (degrees C)

5 heat index (degrees C)
6 dew point (degrees C)
7 relative humidity (%)
8 absolute humidity (g/m^3)
9 station pressure (mbar)
10 barometric pressure (mbar)
11 altitude (m)

Troubleshooting and Support

Support

- Contact us at Vernier Science Education by email at: support@vernier.com
- Go to the Vernier Science Education website (www.vernier.com) and send us a chat message
- Call us at 503 277 2299 or 1 888 VERNIER
- Post a question at: https://github.com/VernierST/godirect-examples/issues
- Ask a question on the glowscript forum at: https://groups.google.com/g/glowscript-users

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 or may not receive some helpful error messages. Otherwise, comment out code to simplify your starter program.
- Try a different browser. In most cases using Chrome is suggested.
- Write a simple Web VPython starter program that does not use Go Direct sensors and does not use the functions described above. This can be a good troubleshooting step if you are not sure why VPython is not launching. Go to https://glowscript.org/ and sign in. Go to your programs and create a new program. Here is an easy example to try:

```
Web VPython 3.2
sphere()
```

- 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 this error:

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

That is a good thing. You probably just need to use the standard Share or export this program and the Download as HTML procedure to use the program.

The Help Built into Web VPython

There is great information on VPython on the internet, starting with the Help built into Web VPython. Just click on the Help button at the top right of the screen. You can then get information on any of the objects you can add to your program, like boxes, spheres, arrows, helixes, etc. You can also learn about canvases.

The Web VPython examples at https://www.glowscript.org/#/user/GlowScriptDemos/folder/Examples/ are just 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.

Other Web Sites for Information on 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:
https://matter-interactions.trinket.io/00_welcome_to_vpython#/welcome-to-vpython/getting-started
https://rhettallain.com/category/python/
https://www.youtube.com/watch?v=8M9q0tydzMA