PYOA for PyGamer/PyBadge - Adding Cursor Support to CircuitPython

Created by Brent Rubell

https://learn.adafruit.com/cursor-for-circuitpython

Last updated on 2021-11-15 07:43:09 PM EST

Table of Contents

Overview

Want to control your CircuitPython game with your PyGamer's Joystick? Click buttons on the display using the buttons on your PyGamer/PyBadge?

We've created a Cursor module for CircuitPython. This module provides an interactive mouse cursor to add to your game or interface.

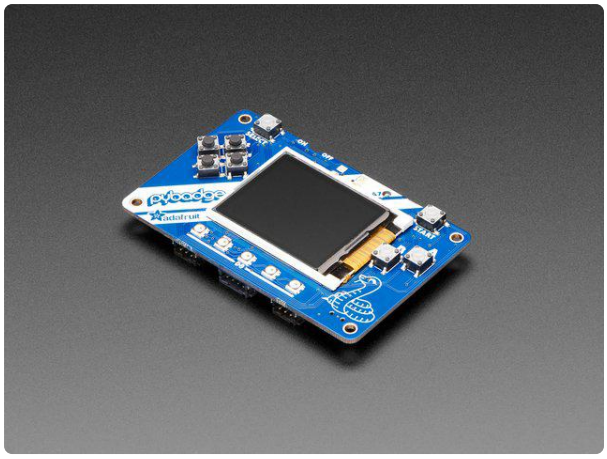
In this guide, you'll learn how to add a cursor to your CircuitPython device's display and control it with your PyBadge or PyGamer.

Then, we added cursor support to the Python Choose your Own Adventure gaming framework (PYOA) so you can play choose-your-own-adventure games on your PyBadge/PyGamer.

Parts

Any of the Adafruit Open-Source gaming platforms will work with this guide:

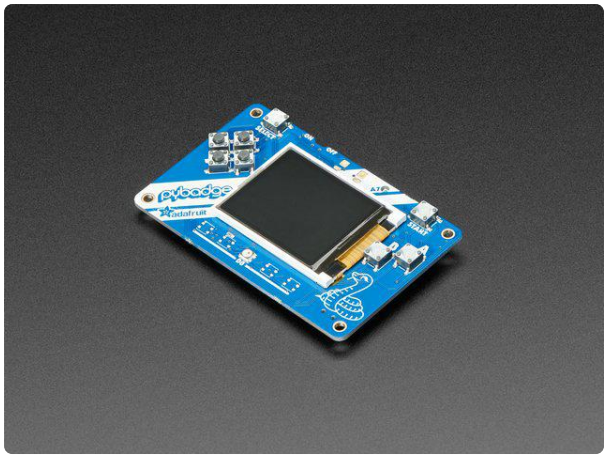
Adafruit PyGamer for MakeCode Arcade, CircuitPython or Arduino
What fits in your pocket, is fully Open Source, and can run CircuitPython, MakeCode Arcade or Arduino games you write yourself? That's right, it's the Adafruit...
https://www.adafruit.com/product/4242



Adafruit PyBadge for MakeCode Arcade, CircuitPython, or Arduino

What's the size of a credit card and can run CircuitPython, MakeCode Arcade or Arduino? That's right, its the Adafruit PyBadge! We wanted to see how much we...

https://www.adafruit.com/product/4200



Adafruit PyBadge LC - MakeCode Arcade, CircuitPython, or Arduino

What's the size of a credit card and can run CircuitPython, MakeCode Arcade or Arduino even when you're on a budget? That's right, it's the Adafruit...

https://www.adafruit.com/product/3939

Adafruit PyGamer Starter Kit

Please note: you may get a royal blue or purple case with your starter kit (they're both lovely colors)What fits in your pocket, is fully Open...

https://www.adafruit.com/product/4277

Materials

1 x USB Cable — https://www.adafruit.com/product/592

USB Cable - USB A to Micro-B

1 x USB Cable — https://www.adafruit.com/product/4148

USB Cable - USB A to micro-B, purple, 2m long

Code Overview

It's easy to use the CursorControl module (https://adafru.it/FbE) with CircuitPython. This module allows you to easily write Python code which generates and controls a mouse cursor on your CircuitPython device's display.

CursorControl Module Installation

Next you'll need to install the Adafruit CircuitPython CursorControl (https://adafru.it/FbE) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/zdx). Our introduction guide has a great page on how to install the library bundle (https://adafru.it/ABU) for both express and non-express boards.

Remember for non-express boards like the, you'll need to manually install the necessary libraries from the bundle:

- adafruit_cursorcontrol.mpy
- cursorcontrol_cursormanager.mpy

You can also download the adafruit_cursorcontrol.mpy from its releases page on Github (https://adafru.it/Fdc).

Before continuing make sure your board's lib folder or root filesystem has the adafrui t_cursorcontrol.mpy and cursorcontrol_cursormanager.mpy files are copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) so you are at the CircuitPython >>> prompt.

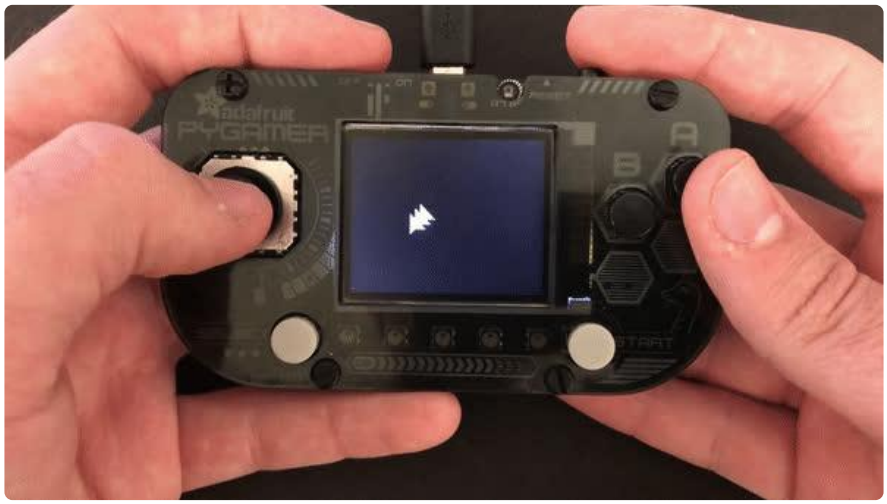
CircuitPython Usage

To demonstrate the usage of the CursorControl library, we'll use the example below. Save the file below to a code.py file. Then, upload the code.py file to the CIRCUITPY drive.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import displayio
from adafruit_cursorcontrol.cursorcontrol import Cursor
from adafruit_cursorcontrol.cursorcontrol_cursormanager import CursorManager

# Create the display
display = board.DISPLAY

# Create the display context
splash = displayio.Group()

# initialize the mouse cursor object
mouse_cursor = Cursor(display, display_group=splash)

# initialize the cursormanager
cursor = CursorManager(mouse_cursor)

# show displayio group
display.show(splash)

while True:
    cursor.update()
    if cursor.is_clicked:
        if mouse_cursor.hidden:
            mouse_cursor.show()
        else:
            mouse_cursor.hide()
    time.sleep(0.01)
```

After saving the code.py to your CIRCUITPY drive, you should see a cursor appear on your device's display.

Move the cursor around by pressing the PyBadge's D-Pad buttons or moving the PyGamer's Joystick!



CursorControl Module Overview

The CursorControl library (https://adafru.it/FbF) contains two classes, `Cursor` and `CursorManager`. The `Cursor` class is responsible for generating a cursor bitmap, displaying it on the screen, and adjusting the cursor's properties.

The `CursorManager` is a high-level class responsible for initializing a hardware interface for the cursor. This includes methods to set up the joystick or buttons and control the cursor's properties using them.

Let's take a look at a short example to understand how these two classes interact.

First, the example creates a display object (using the board's builtin `DISPLAY`) and a displayio group. This step is required - the `Cursor` class does not handle generating a displayio group.

- For more information about DisplayIO Groups in CircuitPython, check out this guide. (https://adafru.it/FbG)

```
# Create the display
display = board.DISPLAY

# Create the display context
splash = displayio.Group(max_size=5)
```

The example next initializes the `Cursor` object by passing it the board's display interface and the display group.

If you want to set the cursor's properties during `Cursor` initialization (to hide the cursor when a game starts or increase the cursor's scale) - take a look at the `__init__` keyword arguments for this class here (https://adafru.it/FbH).

```
# initialize the mouse cursor object
mouse_cursor = Cursor(display, display_group=splash)
```

While you now have a cursor object (and loading the code so far will make a cursor appear on your display) - you need a way of controlling it. You could control it from the loop and create methods to read the joystick and buttons, but we've already gone ahead and built a class to do this.

If you're using a PyGamer or PyBadge, all you need to do to control a `Cursor` is pass the **`Cursor`** object to the **`CursorManager`**

```
# initialize the cursormanager
cursor = CursorManager(mouse_cursor)
```

Then, you'll set the display to "show" the displayio splash group

```
# show displayio group
display.show(splash)
```

The top of the loop calls `cursor.update()`. This method within `CursorManager` handles button presses, joystick movement and d-pad button clicks. It also sets the `CursorManager`'s `is_clicked` property.

```
while True:
    cursor.update()
    if cursor.is_clicked:
        if mouse_cursor.hide:
            mouse_cursor.hide = False
        else:
            mouse_cursor.hide = True
    time.sleep(0.01)
```

Next, we'll want to see if the cursor was clicked. The `.is_clicked` property is set by a call to `cursor.update()` and is a boolean. If it's true, you can hide the `Cursor` by setting its `.hide` property.

Buttons and Text Elements with Cursor

Below is an advanced example which uses the Display Text module as well as the Button module. Moving the cursors over the button and clicking set different Cursor attributes such as cursor speed and cursor scale size.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import displayio
from adafruit_button import Button
from adafruit_display_text import label
import terminalio
from adafruit_cursorcontrol.cursorcontrol import Cursor
from adafruit_cursorcontrol.cursorcontrol_cursormanager import CursorManager

# Create the display
display = board.DISPLAY

# Create the display context
splash = displayio.Group()

# Use the built-in system font
font = terminalio.FONT
```

```
##################################################################################
# Make a background color fill

color_bitmap = displayio.Bitmap(display.width, display.height, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x404040
bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)

##################################################################################

# Set up button/label properties
BUTTON_WIDTH = 80
BUTTON_HEIGHT = 40
BUTTON_MARGIN = 20
LBL_HEADER = [100, 20]
LBL_TEXT = [120, 40]

# Resize buttons for small display (PyGamer)
if display.width < 240:
    BUTTON_WIDTH = int(BUTTON_WIDTH / 2)
    BUTTON_HEIGHT = int(BUTTON_HEIGHT / 2)
    BUTTON_MARGIN = int(BUTTON_MARGIN / 2)
    LBL_HEADER[0] -= 75
    LBL_HEADER[1] -= 10
    LBL_TEXT[0] -= 70
    LBL_TEXT[1] += 55

# Create the buttons
buttons = []

button_speed_inc = Button(
    x=BUTTON_MARGIN,
    y=BUTTON_MARGIN + BUTTON_HEIGHT,
    width=BUTTON_WIDTH,
    height=BUTTON_HEIGHT,
    label="Speed+",
    label_font=font,
)
buttons.append(button_speed_inc)

button_speed_dec = Button(
    x=BUTTON_MARGIN,
    y=BUTTON_MARGIN * 4 + BUTTON_HEIGHT,
    width=BUTTON_WIDTH,
    height=BUTTON_HEIGHT,
    label="Speed-",
    label_font=font,
)
buttons.append(button_speed_dec)

button_scale_pos = Button(
    x=BUTTON_MARGIN * 3 + 2 * BUTTON_WIDTH,
    y=BUTTON_MARGIN + BUTTON_HEIGHT,
    width=BUTTON_WIDTH,
    height=BUTTON_HEIGHT,
    label="Scale+",
    label_font=font,
    style=Button.SHADOWRECT,
)
buttons.append(button_scale_pos)

button_scale_neg = Button(
    x=BUTTON_MARGIN * 3 + 2 * BUTTON_WIDTH,
    y=BUTTON_MARGIN * 4 + BUTTON_HEIGHT,
    width=BUTTON_WIDTH,
    height=BUTTON_HEIGHT,
    label="Scale-",
    label_font=font,
```

```
    style=Button.SHADOWRECT,
)
buttons.append(button_scale_neg)

# Show the button
for b in buttons:
    splash.append(b.group)

# Create a text label
text_label = label.Label(
    font, text="CircuitPython Cursor!", color=0x00FF00, x=LBL_HEADER[0],
y=LBL_HEADER[1]
)
splash.append(text_label)

text_speed = label.Label(font, color=0x00FF00, x=LBL_TEXT[0], y=LBL_TEXT[1])
splash.append(text_speed)

text_scale = label.Label(font, color=0x00FF00, x=LBL_TEXT[0], y=LBL_TEXT[1] + 20)
splash.append(text_scale)

# initialize the mouse cursor object
mouse_cursor = Cursor(display, display_group=splash)

# initialize the cursormanager
cursor = CursorManager(mouse_cursor)

# show displayio group
display.show(splash)

prev_btn = None
while True:
    cursor.update()
    if cursor.is_clicked is True:
        for i, b in enumerate(buttons):
            if b.contains((mouse_cursor.x, mouse_cursor.y)):
                b.selected = True
                print("Button %d pressed" % i)
                if i == 0:  # Increase the cursor speed
                    mouse_cursor.speed += 1
                elif i == 1:  # Decrease the cursor speed
                    mouse_cursor.speed -= 1
                if i == 2:  # Increase the cursor scale
                    mouse_cursor.scale += 1
                elif i == 3:  # Decrease the cursor scale
                    mouse_cursor.scale -= 1
                prev_btn = b
    elif prev_btn is not None:
        prev_btn.selected = False
    text_speed.text = "Speed: {0}px".format(mouse_cursor.speed)
    text_scale.text = "Scale: {0}px".format(mouse_cursor.scale)
    time.sleep(0.1)
```

CursorControl Docs

CursorControl Docs (https://adafru.it/FbB)

Adding Cursor to PYOA

We enjoyed Dave Astel's CircuitPython Choose your Own Adventure guide (https://adafru.it/FbI) - but it required a PyPortal to tap the buttons on the display. We wanted to play PYOA games on our PyBadge/PyGamer, so we added CursorControl to PYOA framework to let you click the buttons instead of tapping them.

If you want to use PYOA on your PyBadge/PyGamer, follow along below!

Before you get started, you'll want to prepare your PyBadge/PyGamer with the required libraries to build a choose your own adventure game:

- Follow the steps listed on this page (https://adafru.it/FbJ) and come back here when you've followed them.

Code Setup

Clicking on download Project zip below will give you a zip file that you will need to dig into to get the cave directory. Copy this directory to your CIRCUITPY drive.

```
[
    {
        "card_id": "start",
        "background_image": "page01.bmp",
        "text": "Welcome adventurer. Your adventure begins, as many do, in Ye Olde
Inn.",
```

```
        "text_color": "0x000001",
        "button01_text": "Continue",
        "button01_goto_card_id": "inn"
    },
    {
        "card_id": "inn",
        "background_image": "page01.bmp",
        "sound": "pub.wav",
        "sound_repeat": "True",
        "text": "This is a peaceful, happy inn with plentiful drink, tasty food,
and friendly staff.",
        "text_color": "0x000001",
        "button01_text": "Stay",
        "button01_goto_card_id": "inn",
        "button02_text": "Go!",
        "button02_goto_card_id": "cave entrance"
    },
    {
        "card_id": "cave entrance",
        "background_image": "page01.bmp",
        "text": "There is a dark cave in the hillside before you.",
        "text_color": "0x000001",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "button01_text": "Enter",
        "button01_goto_card_id": "entry",
        "button02_text": "Run",
        "button02_goto_card_id": "inn"
    },
    {
        "card_id": "entry",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "You are in a dark, narrow tunnel.",
        "text_color": "0x000001",
        "button01_text": "Next",
        "button01_goto_card_id": "side opening"
    },
    {
        "card_id": "side opening",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "You are in a small room, one tunnel leads ahead and another to the
side. Do you continue on or explore the side tunnel?",
        "text_color": "0x000001",
        "button01_text": "Continue",
        "button01_goto_card_id": "skeleton room",
        "button02_text": "Side T.",
        "button02_goto_card_id": "treasure room"
    },
    {
        "card_id": "treasure room",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "There is a pile of treasure here. Congratulations!",
        "text_color": "0x000001",
        "button01_text": "Next",
        "button01_goto_card_id": "maze 1"
    },
    {
        "card_id": "skeleton room",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "There is a skeleton on the floor. From the items around it, it
seems to be that of an unfortunate adventurer.",
```

```
        "text_color": "0x000001",
        "button01_text": "Next",
        "button01_goto_card_id": "maze 2"
    },
    {
        "card_id": "maze 1",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "There are passages to the left and right.",
        "text_color": "0x000001",
        "button01_text": "Left",
        "button01_goto_card_id": "maze 3",
        "button02_text": "Right",
        "button02_goto_card_id": "maze 2",
    },
    {
        "card_id": "maze 2",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "There are passages to the left and right.",
        "text_color": "0x000001",
        "button01_text": "Left",
        "button01_goto_card_id": "maze 1",
        "button02_text": "Right",
        "button02_goto_card_id": "maze 4"
    },
    {
        "card_id": "maze 3",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "There are passages to the left and right.",
        "text_color": "0x000001",
        "button01_text": "Left",
        "button01_goto_card_id": "maze 5",
        "button02_text": "Right",
        "button02_goto_card_id": "maze 2"
    },
    {
        "card_id": "maze 4",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "There are passages to the left and right.",
        "text_color": "0x000001",
        "button01_text": "Left",
        "button01_goto_card_id": "maze 1",
        "button02_text": "Right",
        "button02_goto_card_id": "maze 6"
    },
    {
        "card_id": "maze 5",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
        "text": "There are passages to the left and right.",
        "text_color": "0x000001",
        "button01_text": "Left",
        "button01_goto_card_id": "maze 4",
        "button02_text": "Right",
        "button02_goto_card_id": "creaking"
    },
    {
        "card_id": "maze 6",
        "background_image": "page01.bmp",
        "sound": "cave.wav",
        "sound_repeat": "True",
```

```
        "text": "There are passages to the left and right.",
        "text_color": "0x000001",
        "button01_text": "Left",
        "button01_goto_card_id": "creaking",
        "button02_text": "Right",
        "button02_goto_card_id": "maze 3"
    },
    {
        "card_id": "creaking",
        "background_image": "page01.bmp",
        "sound": "creak.wav",
        "sound_repeat": "True",
        "text": "You hear an ominuous creaking from around the corner",
        "text_color": "0x000001",
        "button01_text": "Cont.",
        "button01_goto_card_id": "bridge room 1",
        "button02_text": "Go back",
        "button02_goto_card_id": "inn"
    },
    {
        "card_id": "bridge room 1",
        "background_image": "page01.bmp",
        "sound": "creak.wav",
        "sound_repeat": "True",
        "text": "There is a creaking, rickity wooded bridge leading across a gaping
chasm.",
        "text_color": "0x000001",
        "button01_text": "Cont.",
        "button01_goto_card_id": "bridge room 2"
    },

    {
      "card_id": "bridge room 2",
      "background_image": "page01.bmp",
      "sound": "creak.wav",
      "sound_repeat": "True",
      "text": "At the other end is a large treasure chest. There is also a short
tunnel with daylight at the end.",
      "text_color": "0x000001",
      "button01_text": "Treasure!",
      "button01_goto_card_id": "die",
      "button02_text": "Leave",
      "button02_goto_card_id": "inn"
    },

    {
        "card_id": "die",
        "background_image": "page01.bmp",
        "sound": "scream.wav",
        "text": "The bridge gives way and you fall to a painful death.",
        "text_color": "0x000001",
        "button01_text": "Next",
        "button01_goto_card_id": "inn"
    }
]
```

In the `code.py` file on your device, set the example as the cave example. You can do this by changing the following line from:

`gfx.load_game("/cyoa")`

to

```
gfx.load_game("/cave")
```

Code Overview

Your PyGamer/PyBadge will automatically load the cave game and create a cursor overlay. You can move the cursor around using the joystick on the PyGamer or the D-Pad buttons on the PyBadge.

To click the buttons on the display, press the A button. This is the same button for both the PyBadge and the PyGamer.

That's it! If you would like to try your hand at designing or writing your own adventure game, check out this page on Dave Astel's guide (https://adafru.it/FbK).