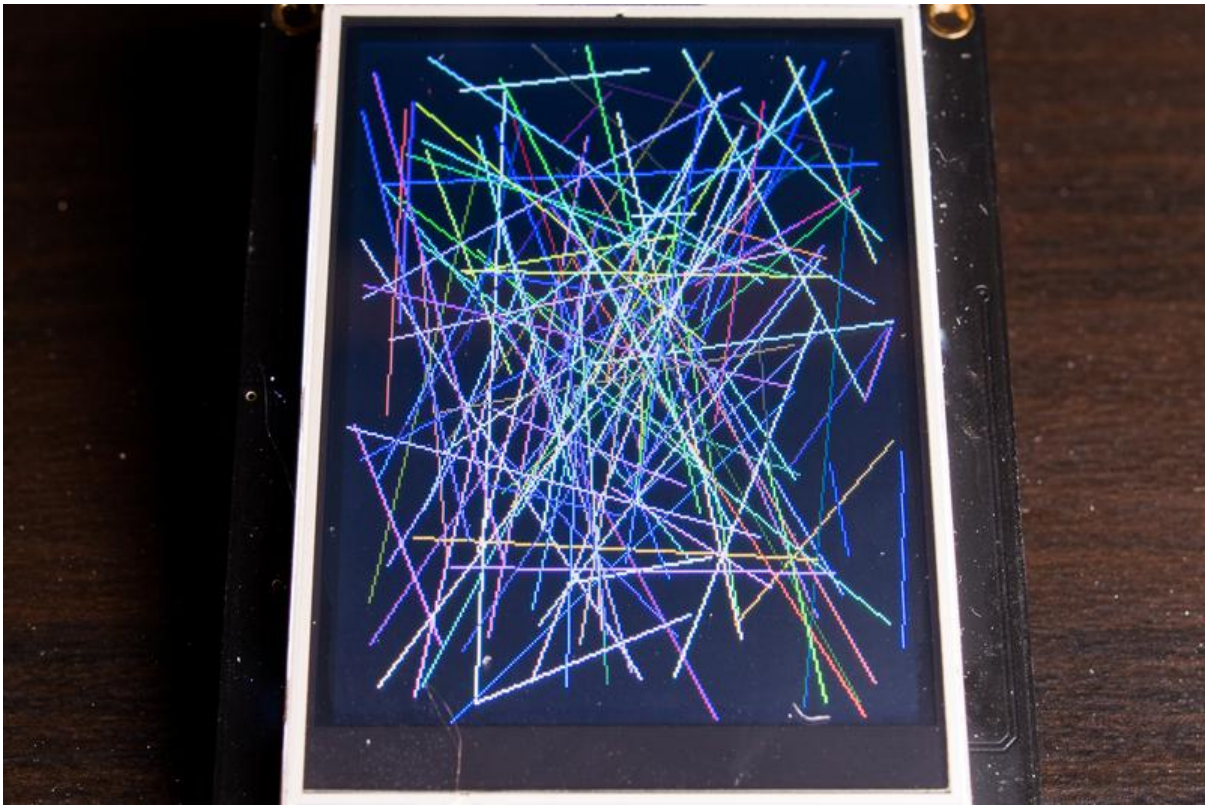
MicroPython Displays: Drawing Shapes

Created by Tony DiCola



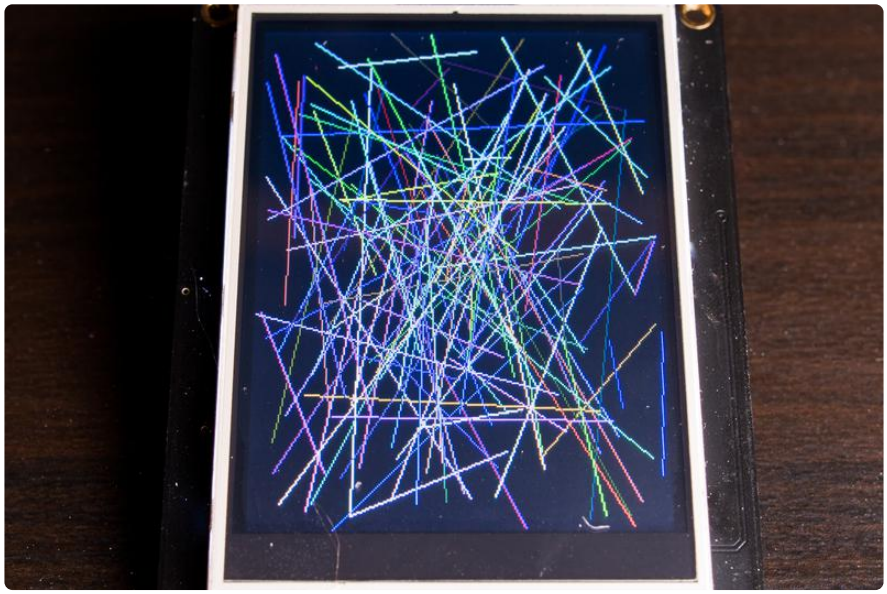
https://learn.adafruit.com/micropython-displays-drawing-shapes

Last updated on 2021-11-15 06:49:50 PM EST

Table of Contents

Overview

Note this guide was written for MicroPython.org firmware and not Adafruit CircuitPython firmware.



So you have a nifty display like a bright TFT LCD, Charlieplex LED or NeoPixel matrix powered by MicroPython, but what if you want to draw shapes and graphics on it? Most MicroPython display modules only give you basic pixel drawing commands so it's difficult to draw graphics yourself pixel by pixel. Luckily there's a handy new MicroPython graphics module (https://adafru.it/sBf) you can use to draw basic shapes and graphics on any pixel-based display! This module shows some of the power of MicroPython--a single module can work with any pixel display because of MicroPython's dynamic language features. In this guide you'll learn how to use a MicroPython graphics module to draw basic line, rectangle, circle, and triangle shapes on pixel-based displays like the ILI9341 TFT FeatherWing (https://adafru.it/szd).

To follow this guide you'll want to be familiar with MicroPython by reading these guides:

- MicroPython Basics: What is MicroPython? (https://adafru.it/pXa)
- MicroPython Basics: How to Load MicroPython on a Board (https://adafru.it/pNB)
- MicroPython Basics: Load Files & Run Code (https://adafru.it/s1f)

See all the MicroPython guides in the learning system (https://adafru.it/qzD) for more information.

In addition be sure to follow the guide for your pixel-based display first:

- MicroPython Hardware: LED Backpacks & FeatherWings (https://adafru.it/spc)
- MicroPython Hardware: Charlieplex LED Matrix (https://adafru.it/spb)
- MicroPython Hardware: SSD1306 OLED Display (https://adafru.it/spd)
- MicroPython Hardware: ILI9341 TFT & FeatherWing (https://adafru.it/sB4)

Hardware

For this guide there's no special hardware you need to draw graphics on a display. However you do need to have a pixel-based display of some sort (LED, TFT, NeoPixel--anything!) connected to your MicroPython board. Check out the following guides for details on how to use a few pixel displays with MicroPython:

- MicroPython Hardware: LED Backpacks & FeatherWings (https://adafru.it/spc)
- MicroPython Hardware: Charlieplex LED Matrix (https://adafru.it/spb)
- MicroPython Hardware: SSD1306 OLED Display (https://adafru.it/spd)
- MicroPython Hardware: ILI9341 TFT & FeatherWing (https://adafru.it/sB4)

Follow the appropriate guide to setup the hardware and make sure you can draw pixels on the display with MicroPython before continuing.

Software

Install Module

To use the graphics module with your MicroPython board you'll need to install the micropython-adafruit-gfx MicroPython module (https://adafru.it/sBf) on your board.

First make sure you are running the latest version of MicroPython for your board. If you're using the ESP8266 MicroPython port you must be running version 1.8.5 or higher (https://adafru.it/sas) as earlier versions do not support using .mpy modules as shown in this guide.

Download the latest gfx.mpy file from the releases page (https://adafru.it/sBg) of the micropython-adafruit-gfx GitHub repository (https://adafru.it/sBh).

If your board supports USB mass storage, like the SAMD21 MicroPython port, then simply drag the .mpy and other files to the board's file system (eject the drive and reset the board to make sure it is picked up by MicroPython).

If your board doesn't support USB mass storage, like ESP8266 MicroPython boards, then use a tool like ampy to copy the file to the board (https://adafru.it/r2B).

Usage

The following section will show how to draw shapes on a ILI9341 TFT display like the TFT FeatherWing. You'll see how the module can be adpated to draw on any pixel-based display by plugging in a new pixel drawing function too.

First connect to the board's serial REPL (https://adafru.it/pMf)so you are at the MicroPython >>> prompt.

Display Initialization

Next you'll need to initialize your display so that you can draw pixels on it. Consult the MicroPython display guides (https://adafru.it/sBi) for details on initializing displays.

For example the ILI9341 TFT FeatherWing display initialization on ESP8266 MicroPython might look like:

```
import machine
import ili9341
spi = machine.SPI(1, baudrate=32000000)
display = ili9341.ILI9341(spi, cs=machine.Pin(0), dc=machine.Pin(15))
```

Once you have a display object which has a function to draw pixels you're ready to start drawing shapes.

Graphics Initialization

To use the graphics and shape rendering module you'll need to import its module and create an instance of the GFX class inside it. For example to create a graphics renderer for the TFT FeatherWing above you would run:

```
import gfx
graphics = gfx.GFX(240, 320, display.pixel)
```

The GFX class initializer takes three parameters:

1. The maximum width of the display in pixels, in this case 240 for the TFT FeatherWing.
2. The maximum height of the display in pixels, in this case 320 for the TFT FeatherWing.
3. A pixel drawing function to call when the GFX class needs to write a pixel. For this example the TFT display class pixel function is specified. It's important to note that to use the graphics class you must have a function it can call to draw pixels! This function can live anywhere, like as a global function or on a class instance. The function needs to take at least a pixel x position and pixel y position parameter (in that order), and any number of other positional and keyword parameters after them (like color, intensity, etc.).

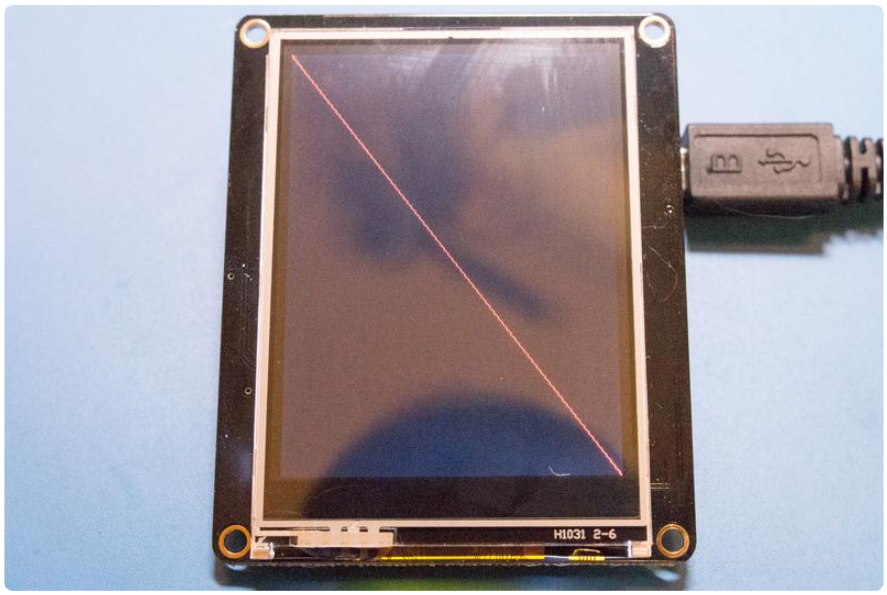
There are two optional parameters you can specify as keyword arguments too. They aren't shown in this guide but see how the ILI9341 display example code (https://adafru.it/sBj)uses them:

- hline - Optionally set hline to a fast horizontal line drawing function for your display. This will improve the speed of drawing certain shapes. The function should take as parameters an x position and y position of the line start, then width of the line in pixels. Any number of optional color or other parameters can follow. If you don't provide a hline function a slow default implementation that draws pixel by pixel will be used.
- vline - Optionally set vline to a fast vertical line drawing function for your display. This will improve the speed of drawing certain shapes, especially filled shapes. The function should take as parameters an x position, y position of the line start, then height of the line in pixels. Any number of optional color or other parameters can follow. If you don't provide a vline function a slow default implementation that draws pixel by pixel will be used.

Drawing Shapes

Now the fun begins! After creating the GFX class you can call functions to draw shapes on the display. For example draw a simple line across the entire display with the line function:

```
display.fill(0)  # Clear the display
graphics.line(0, 0, 239, 319, ili9341.color565(255, 0, 0))
```

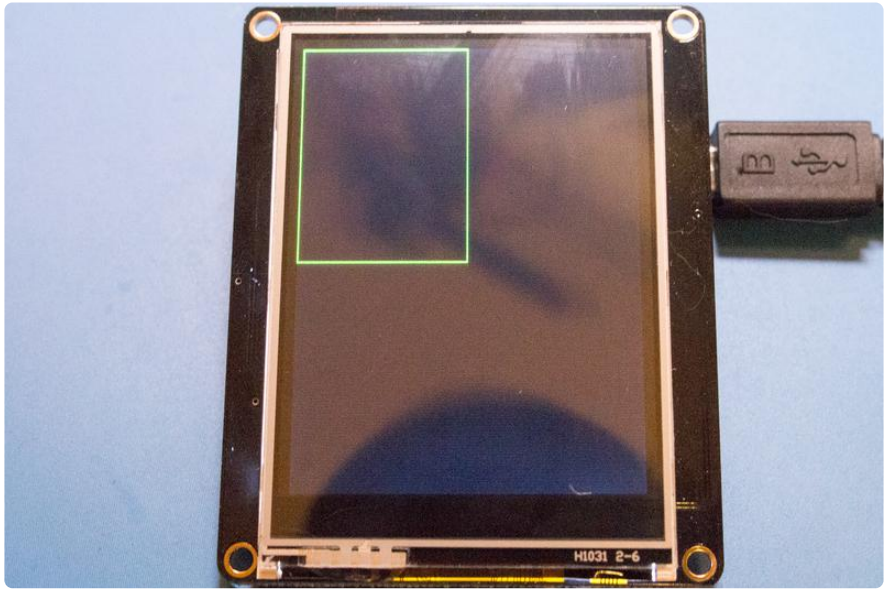


The line function takes the following parameters:

- X position of the line start.
- Y position of the line start.
- X position of the line end.
- Y position of the line end.
- Any number of color or other parameters that will be passed to the pixel drawing function. In this case a red color for the display is specified.

Draw an empty rectangle with the rect function:

```
display.fill(0)  # Clear the display
graphics.rect(0, 0, 120, 160, ili9341.color565(0, 255, 0))
```

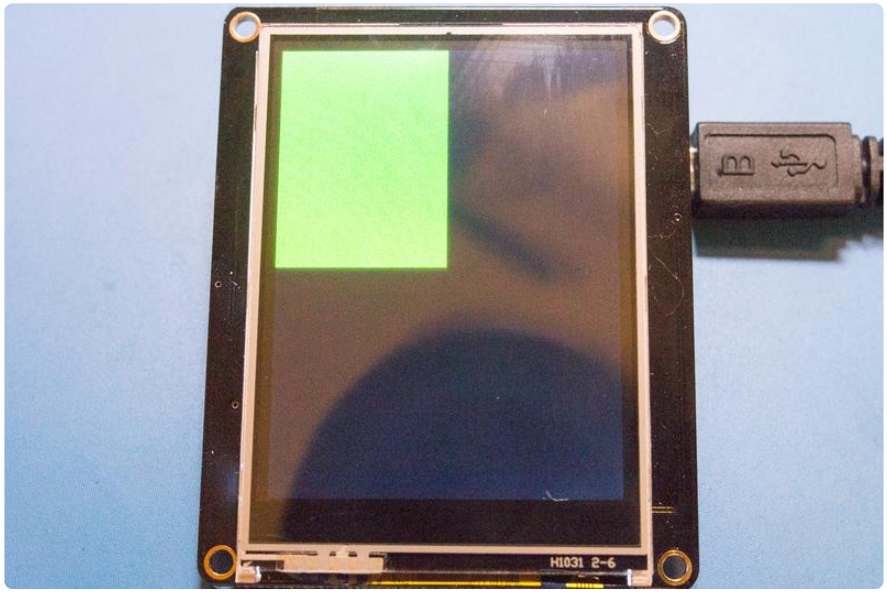


The rect function takes the following parameters:

- X position of the rectangle upper left corner.
- Y position of the rectangle upper left corner.
- Width of the rectangle in pixels.
- Height of the rectangle in pixels.
- Any number of color other parameters that will be passed to the pixel drawing function. In this case a green color for the display is specified.

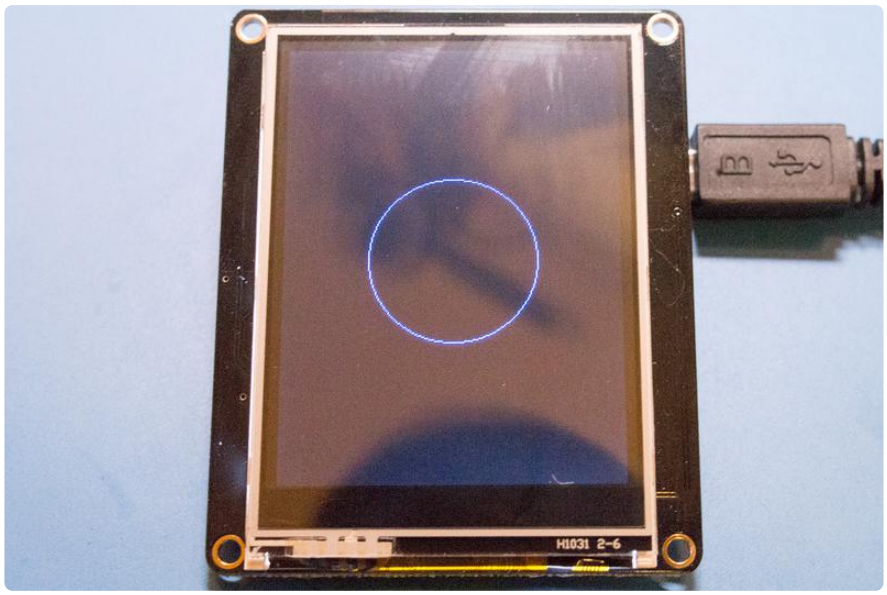
You can also draw a filled rectangle with the fill_rect function. The parameters are exactly the same as the rect function but now the rectangle is drawn as a solid shape:

```
display.fill(0)  # Clear the display
graphics.fill_rect(0, 0, 120, 160, ili9341.color565(0, 255, 0))
```



Draw a circle with the circle function:

```
display.fill(0)  # Clear the display.
graphics.circle(120, 160, 60, ili9341.color565(0, 0, 255))
```



The circle function takes the following parameters:

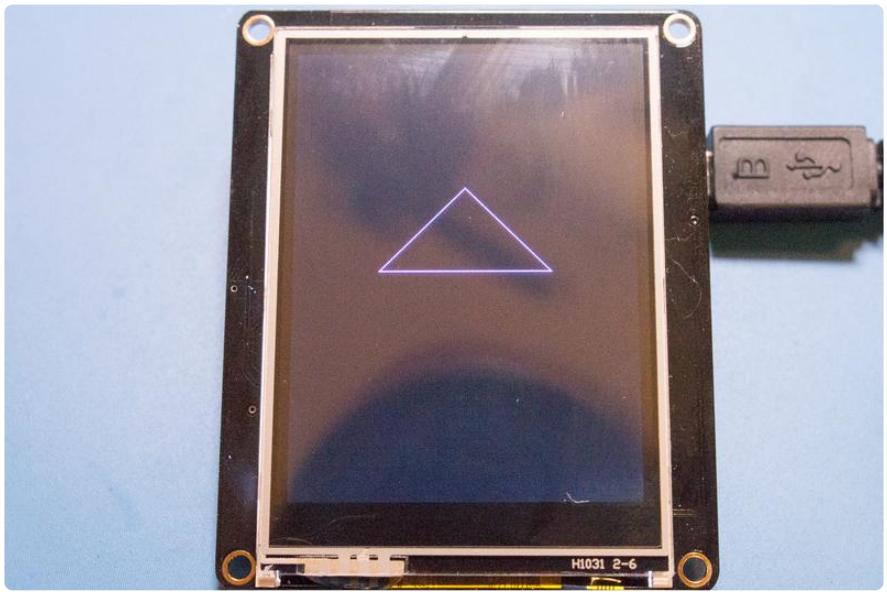
- X position of the circle center.
- Y position of the circle center.
- Radius of the circle in pixels.
- Any number of color other parameters that will be passed to the pixel drawing function. In this case a blue color for the display is specified.

It's no surprise there's a fill_circle function to draw a filled circle. This function takes exactly the same parameters as the circle function but will draw a solid shape:

```
display.fill(0)  # Clear the display
graphics.fill_circle(120, 160, 60, ili9341.color565(0, 0, 255))
```

Draw a triangle with the triangle function:

```
display.fill(0)  # Clear the display
graphics.triangle(120, 100, 180, 160, 60, 160, ili9341.color565(255, 0, 255))
```

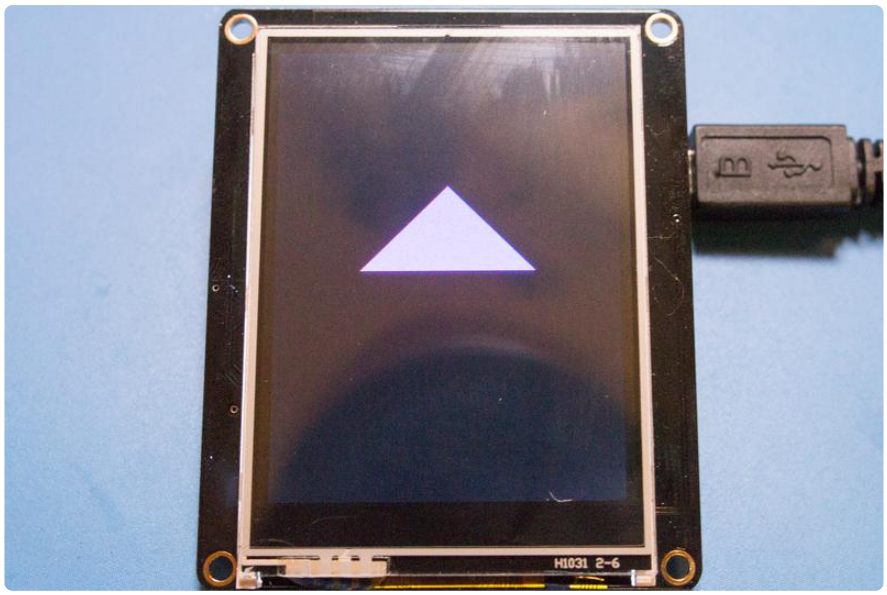


The triangle function takes the following parameters:

- X position of the first point in the triangle.
- Y position of the first point in the triangle.
- X position of the second point in the triangle.
- Y position of the second point in the triangle.
- X position of the third point in the triangle.
- Y position of the third point in the triangle.
- Any number of color other parameters that will be passed to the pixel drawing function. In this case a pink color for the display is specified.

And a fill_triangle function exists to draw a filled triangle too. This function takes exactly the same parameters as the triangle function but will draw a solid shape:

```
display.fill(0)  # Clear the display
graphics.fill_triangle(120, 100, 180, 160, 60, 160, ili9341.color565(255, 0, 255))
```



That's all there is to basic shape drawing with the Adafruit MicroPython GFX module! With these basic shape primitives you can start to create interesting graphic projects, like a pong or breakout game using filled rectangles and circles. You can add text with the Adafruit MicroPython bitmap font module (https://adafru.it/sA0) too!

Remember any pixel-based display can be used with this library, for example an OLED display or NeoPixel, Charlieplex LED, or simple LED backpack matrix are great targets to use with the library. Just plug in each module's pixel function and you'll be drawing shapes in no time!