
AdafruitDisplay *shapesLibraryDocumentation*
Release 1.0

Limor Fried

Oct 25, 2021

Contents

1	Dependencies	3
1.1	Installing from PyPI	3
2	Usage Example	5
3	Documentation	7
4	Contributing	9
5	Building locally	11
5.1	Zip release files	11
5.2	Sphinx documentation	11
6	Table of Contents	13
6.1	Simple test	13
6.2	Simple test MagTag	14
6.3	Sparkline Simple Test	16
6.4	Sparkline Ticks Example	18
6.5	Sparkline Triple Test	22
6.6	Circle Animation	27
6.7	circle	28
6.7.1	Implementation Notes	28
6.8	rect	29
6.8.1	Implementation Notes	29
6.9	roundrect	29
6.10	triangle	30
6.10.1	Implementation Notes	30
6.11	line	31
6.11.1	Implementation Notes	31
6.12	polygon	31
6.12.1	Implementation Notes	31
6.13	sparkline	31
6.13.1	Implementation Notes	32
7	Indices and tables	33
	Python Module Index	35

Various common shapes for use with displayio

This driver depends on:

- [Adafruit CircuitPython](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).

1.1 Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally [from PyPI](#). To install for current user:

```
pip3 install adafruit-circuitpython-display_shapes
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-display_shapes
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-display_shapes
```


CHAPTER 2

Usage Example

```
import board
import displayio
from adafruit_display_shapes.rect import Rect
from adafruit_display_shapes.circle import Circle
from adafruit_display_shapes.roundrect import RoundRect

splash = displayio.Group()
board.DISPLAY.show(splash)

color_bitmap = displayio.Bitmap(320, 240, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0xFFFFFF
bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, position=(0, 0))
print(bg_sprite.position)
splash.append(bg_sprite)

triangle = Triangle(170, 50, 120, 140, 210, 160, fill=0x00FF00, outline=0xFF00FF)
splash.append(triangle)

rect = Rect(80, 20, 41, 41, fill=0x0)
splash.append(rect)

circle = Circle(100, 100, 20, fill=0x00FF00, outline=0xFF00FF)
splash.append(circle)

rect2 = Rect(50, 100, 61, 81, outline=0x0, stroke=3)
splash.append(rect2)

roundrect = RoundRect(10, 10, 61, 81, 10, fill=0x0, outline=0xFF00FF, stroke=6)
splash.append(roundrect)

while True:
    pass
```


CHAPTER 3

Documentation

API documentation for this library can be found on [Read the Docs](#).

CHAPTER 4

Contributing

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

5.1 Zip release files

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-display_shapes --
↳library_location .
```

5.2 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/display_shapes_simpletest.py

```
1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 import board
5 import displayio
6 from adafruit_display_shapes.rect import Rect
7 from adafruit_display_shapes.circle import Circle
8 from adafruit_display_shapes.roundrect import RoundRect
9 from adafruit_display_shapes.triangle import Triangle
10 from adafruit_display_shapes.line import Line
11 from adafruit_display_shapes.polygon import Polygon
12
13 # Make the display context
14 splash = displayio.Group()
15 board.DISPLAY.show(splash)
16
17 # Make a background color fill
18 color_bitmap = displayio.Bitmap(320, 240, 1)
19 color_palette = displayio.Palette(1)
20 color_palette[0] = 0xFFFFFF
21 bg_sprite = displayio.TileGrid(color_bitmap, x=0, y=0, pixel_shader=color_palette)
22 splash.append(bg_sprite)
23 #####
24
25 splash.append(Line(220, 130, 270, 210, 0xFF0000))
26 splash.append(Line(270, 210, 220, 210, 0xFF0000))
27 splash.append(Line(220, 210, 270, 130, 0xFF0000))
```

(continues on next page)

(continued from previous page)

```
28 splash.append(Line(270, 130, 220, 130, 0xFF0000))
29
30 # Draw a blue star
31 polygon = Polygon(
32     [
33         (255, 40),
34         (262, 62),
35         (285, 62),
36         (265, 76),
37         (275, 100),
38         (255, 84),
39         (235, 100),
40         (245, 76),
41         (225, 62),
42         (248, 62),
43     ],
44     outline=0x0000FF,
45 )
46 splash.append(polygon)
47
48 triangle = Triangle(170, 50, 120, 140, 210, 160, fill=0x00FF00, outline=0xFF00FF)
49 splash.append(triangle)
50
51 rect = Rect(80, 20, 41, 41, fill=0x0)
52 splash.append(rect)
53
54 circle = Circle(100, 100, 20, fill=0x00FF00, outline=0xFF00FF)
55 splash.append(circle)
56
57 rect2 = Rect(50, 100, 61, 81, outline=0x0, stroke=3)
58 splash.append(rect2)
59
60 roundrect = RoundRect(10, 10, 61, 81, 10, fill=0x0, outline=0xFF00FF, stroke=6)
61 splash.append(roundrect)
62
63
64 while True:
65     pass
```

6.2 Simple test MagTag

Simple test with the MagTag.

Listing 2: examples/display_shapes_simpletest_magtag.py

```
1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 import board
5 import displayio
6 from adafruit_display_shapes.rect import Rect
7 from adafruit_display_shapes.circle import Circle
8 from adafruit_display_shapes.roundrect import RoundRect
9 from adafruit_display_shapes.triangle import Triangle
```

(continues on next page)

(continued from previous page)

```

10 from adafruit_display_shapes.line import Line
11 from adafruit_display_shapes.polygon import Polygon
12
13 # use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
14 # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.
15 # ↪ https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-
16 # ↪ and-display-bus
17 display = board.DISPLAY
18
19 # Make the display context
20 splash = displayio.Group()
21 display.show(splash)
22
23 # Make a background color fill
24 color_bitmap = displayio.Bitmap(display.width, display.height, 1)
25 color_palette = displayio.Palette(1)
26 color_palette[0] = 0xFFFFFF
27 bg_sprite = displayio.TileGrid(color_bitmap, x=0, y=0, pixel_shader=color_palette)
28 splash.append(bg_sprite)
29 #####
30 splash.append(Line(5, 74, 10, 110, 0x000000))
31 splash.append(Line(15, 74, 20, 110, 0x000000))
32 splash.append(Line(25, 74, 30, 110, 0x000000))
33 splash.append(Line(35, 74, 40, 110, 0x000000))
34
35 # Draw star
36 polygon = Polygon(
37     [
38         (255, 40),
39         (262, 62),
40         (285, 62),
41         (265, 76),
42         (275, 100),
43         (255, 84),
44         (235, 100),
45         (245, 76),
46         (225, 62),
47         (248, 62),
48     ],
49     outline=0x000000,
50 )
51 splash.append(polygon)
52
53 triangle = Triangle(170, 20, 140, 90, 210, 100, fill=0x999999, outline=0x000000)
54 splash.append(triangle)
55
56 rect = Rect(80, 20, 41, 41, fill=0x999999, outline=0x666666)
57 splash.append(rect)
58
59 circle = Circle(100, 100, 20, fill=0xFFFFFF, outline=0x000000)
60 splash.append(circle)
61
62 rect2 = Rect(70, 85, 61, 30, outline=0x0, stroke=3)
63 splash.append(rect2)
64

```

(continues on next page)

(continued from previous page)

```
65 roundrect = RoundRect(10, 10, 61, 51, 10, fill=0x666666, outline=0x000000, stroke=6)
66 splash.append(roundrect)
67
68 display.refresh()
69 while True:
70     pass
```

6.3 Sparkline Simple Test

Simple test with Sparklines

Listing 3: examples/display_shapes_sparkline_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  # class of sparklines in CircuitPython
5  # created by Kevin Matocha - Copyright 2020 (C)
6
7  # See the bottom for a code example using the `sparkline` Class.
8
9  # # File: display_shapes_sparkline.py
10 # A sparkline is a scrolling line graph, where any values added to sparkline using
11 # `add_value` are plotted.
12 #
13 # The `sparkline` class creates an element suitable for adding to the display using
14 # `display.show(mySparkline)`
15 # or adding to a `displayio.Group` to be displayed.
16 #
17 # When creating the sparkline, identify the number of `max_items` that will be
18 # included in the graph.
19 # When additional elements are added to the sparkline and the number of items has
20 # exceeded max_items, any excess values are removed from the left of the graph,
21 # and new values are added to the right.
22
23
24 # The following is an example that shows the
25
26 # setup display
27 # instance sparklines
28 # add to the display
29 # Loop the following steps:
30 #     add new values to sparkline `add_value`
31 #     update the sparklines `update`
32
33 import time
34 import random
35 import board
36 import displayio
37
38
39 from adafruit_display_shapes.sparkline import Sparkline
40
41 if "DISPLAY" not in dir(board):
```

(continues on next page)

(continued from previous page)

```

42  # Setup the LCD display with driver
43  # You may need to change this to match the display driver for the chipset
44  # used on your display
45  from adafruit_ili9341 import ILI9341
46
47  displayio.release_displays()
48
49  # setup the SPI bus
50  spi = board.SPI()
51  tft_cs = board.D9 # arbitrary, pin not used
52  tft_dc = board.D10
53  tft_backlight = board.D12
54  tft_reset = board.D11
55
56  while not spi.try_lock():
57      spi.configure(baudrate=32000000)
58
59  spi.unlock()
60
61  display_bus = displayio.FourWire(
62      spi,
63      command=tft_dc,
64      chip_select=tft_cs,
65      reset=tft_reset,
66      baudrate=32000000,
67      polarity=1,
68      phase=1,
69  )
70
71  print("spi.frequency: {}".format(spi.frequency))
72
73  # Number of pixels in the display
74  DISPLAY_WIDTH = 320
75  DISPLAY_HEIGHT = 240
76
77  # create the display
78  display = ILI9341(
79      display_bus,
80      width=DISPLAY_WIDTH,
81      height=DISPLAY_HEIGHT,
82      rotation=180, # The rotation can be adjusted to match your configuration.
83      auto_refresh=True,
84      native_frames_per_second=90,
85  )
86
87  # reset the display to show nothing.
88  display.show(None)
89  else:
90      # built-in display
91      display = board.DISPLAY
92
93  #####
94  # Create background bitmaps and sparklines
95  #####
96
97  # Baseline size of the sparkline chart, in pixels.
98  chart_width = display.width

```

(continues on next page)

(continued from previous page)

```

99 chart_height = display.height
100
101 # sparkline1 uses a vertical y range between 0 to 10 and will contain a
102 # maximum of 40 items
103 sparkline1 = Sparkline(
104     width=chart_width, height=chart_height, max_items=40, y_min=0, y_max=10, x=0, y=0
105 )
106
107 # Create a group to hold the sparkline and append the sparkline into the
108 # group (my_group)
109 #
110 # Note: In cases where display elements will overlap, then the order the elements
111 # are added to the group will set which is on top. Latter elements are displayed
112 # on top of former elements.
113 my_group = displayio.Group()
114
115 # add the sparkline into my_group
116 my_group.append(sparkline1)
117
118
119 # Add my_group (containing the sparkline) to the display
120 display.show(my_group)
121
122 # Start the main loop
123 while True:
124
125     # turn off the auto_refresh of the display while modifying the sparkline
126     display.auto_refresh = False
127
128     # add_value: add a new value to a sparkline
129     # Note: The y-range for mySparkline1 is set to 0 to 10, so all these random
130     # values (between 0 and 10) will fit within the visible range of this sparkline
131     sparkline1.add_value(random.uniform(0, 10))
132
133     # turn the display auto_refresh back on
134     display.auto_refresh = True
135
136     # The display seems to be less jittery if a small sleep time is provided
137     # You can adjust this to see if it has any effect
138     time.sleep(0.01)

```

6.4 Sparkline Ticks Example

Example using tick with the Sparkline class

Listing 4: examples/display_shapes_sparkline_ticks.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 # class of sparklines in CircuitPython
5 # created by Kevin Matocha - Copyright 2020 (C)
6
7 # See the bottom for a code example using the `sparkline` Class.

```

(continues on next page)

(continued from previous page)

```

8
9  # # File: display_shapes_sparkline.py
10 # A sparkline is a scrolling line graph, where any values added to sparkline
11 # using `add_value` are plotted.
12 #
13 # The `sparkline` class creates an element suitable for adding to the display
14 # using `display.show(mySparkline)` or adding to a `displayio.Group` to be displayed.
15 #
16 # When creating the sparkline, identify the number of `max_items` that will be
17 # included in the graph.
18 # When additional elements are added to the sparkline and the number of items
19 # has exceeded max_items, any excess values are removed from the left of the
20 # graph, and new values are added to the right.
21
22
23 # The following is an example that shows the
24
25 # setup display
26 # instance sparklines
27 # add to the display
28 # Loop the following steps:
29 #     add new values to sparkline `add_value`
30 #     update the sparklines `update`
31
32 import random
33 import time
34 import board
35 import displayio
36 import terminalio
37 from adafruit_display_text import label
38 from adafruit_display_shapes.sparkline import Sparkline
39 from adafruit_display_shapes.line import Line
40 from adafruit_display_shapes.rect import Rect
41
42
43 if "DISPLAY" not in dir(board):
44     # Setup the LCD display with driver
45     # You may need to change this to match the display driver for the chipset
46     # used on your display
47     from adafruit_ili9341 import ILI9341
48
49     displayio.release_displays()
50
51     # setup the SPI bus
52     spi = board.SPI()
53     tft_cs = board.D9 # arbitrary, pin not used
54     tft_dc = board.D10
55     tft_backlight = board.D12
56     tft_reset = board.D11
57
58     while not spi.try_lock():
59         spi.configure(baudrate=32000000)
60     spi.unlock()
61
62     display_bus = displayio.FourWire(
63         spi,
64         command=tft_dc,

```

(continues on next page)

(continued from previous page)

```

65     chip_select=tft_cs,
66     reset=tft_reset,
67     baudrate=32000000,
68     polarity=1,
69     phase=1,
70 )
71
72 print("spi.frequency: {}".format(spi.frequency))
73
74 # Number of pixels in the display
75 DISPLAY_WIDTH = 320
76 DISPLAY_HEIGHT = 240
77
78 # create the display
79 display = ILI9341(
80     display_bus,
81     width=DISPLAY_WIDTH,
82     height=DISPLAY_HEIGHT,
83     rotation=180, # The rotation can be adjusted to match your configuration.
84     auto_refresh=True,
85     native_frames_per_second=90,
86 )
87
88 # reset the display to show nothing.
89 display.show(None)
90 else:
91     # built-in display
92     display = board.DISPLAY
93
94 #####
95 # Create background bitmaps and sparklines
96 #####
97
98 # Baseline size of the sparkline chart, in pixels.
99 chart_width = display.width - 50
100 chart_height = display.height - 50
101
102 font = terminalio.FONT
103
104 line_color = 0xFFFFFF
105
106 # Setup the first bitmap and sparkline
107 # This sparkline has no background bitmap
108 # mySparkline1 uses a vertical y range between 0 to 10 and will contain a
109 # maximum of 40 items
110 sparkline1 = Sparkline(
111     width=chart_width,
112     height=chart_height,
113     max_items=40,
114     y_min=0,
115     y_max=10,
116     x=40,
117     y=30,
118     color=line_color,
119 )
120
121 # Label the y-axis range

```

(continues on next page)

(continued from previous page)

```

122
123 text_xoffset = -10
124 text_labella = label.Label(
125     font=font, text=str(sparkline1.y_top), color=line_color
126 ) # yTop label
127 text_labella.anchor_point = (1, 0.5) # set the anchorpoint at right-center
128 text_labella.anchored_position = (
129     sparkline1.x + text_xoffset,
130     sparkline1.y,
131 ) # set the text anchored position to the upper right of the graph
132
133 text_labellb = label.Label(
134     font=font, text=str(sparkline1.y_bottom), color=line_color
135 ) # yTop label
136 text_labellb.anchor_point = (1, 0.5) # set the anchorpoint at right-center
137 text_labellb.anchored_position = (
138     sparkline1.x + text_xoffset,
139     sparkline1.y + chart_height,
140 ) # set the text anchored position to the upper right of the graph
141
142
143 bounding_rectangle = Rect(
144     sparkline1.x, sparkline1.y, chart_width, chart_height, outline=line_color
145 )
146
147
148 # Create a group to hold the sparkline, text, rectangle and tickmarks
149 # append them into the group (my_group)
150 #
151 # Note: In cases where display elements will overlap, then the order the
152 # elements are added to the group will set which is on top. Latter elements
153 # are displayed on top of former elemtns.
154
155 my_group = displayio.Group()
156
157 my_group.append(sparkline1)
158 my_group.append(text_labella)
159 my_group.append(text_labellb)
160 my_group.append(bounding_rectangle)
161
162 total_ticks = 10
163
164 for i in range(total_ticks + 1):
165     x_start = sparkline1.x - 5
166     x_end = sparkline1.x
167     y_both = int(round(sparkline1.y + (i * (chart_height) / (total_ticks))))
168     if y_both > sparkline1.y + chart_height - 1:
169         y_both = sparkline1.y + chart_height - 1
170     my_group.append(Line(x_start, y_both, x_end, y_both, color=line_color))
171
172
173 # Set the display to show my_group that contains the sparkline and other graphics
174 display.show(my_group)
175
176 # Start the main loop
177 while True:
178

```

(continues on next page)

(continued from previous page)

```

179 # Turn off auto_refresh to prevent partial updates of the screen during updates
180 # of the sparkline drawing
181 display.auto_refresh = False
182
183 # add_value: add a new value to a sparkline
184 # Note: The y-range for mySparkline1 is set to 0 to 10, so all these random
185 # values (between 0 and 10) will fit within the visible range of this sparkline
186 sparkline1.add_value(random.uniform(0, 10))
187
188 # Turn on auto_refresh for the display
189 display.auto_refresh = True
190
191 # The display seems to be less jittery if a small sleep time is provided
192 # You can adjust this to see if it has any effect
193 time.sleep(0.01)

```

6.5 Sparkline Triple Test

reate background bitmaps and sparklines

Listing 5: examples/display_shapes_sparkline_triple.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 # class of sparklines in CircuitPython
5 # created by Kevin Matocha - Copyright 2020 (C)
6
7 # See the bottom for a code example using the `sparkline` Class.
8
9 # # File: display_shapes_sparkline.py
10 # A sparkline is a scrolling line graph, where any values added to sparkline
11 # using `add_value` are plotted.
12 #
13 # The `sparkline` class creates an element suitable for adding to the display
14 # using `display.show(mySparkline)` or adding to a `displayio.Group` to be displayed.
15 #
16 # When creating the sparkline, identify the number of `max_items` that will be
17 # included in the graph.
18 # When additional elements are added to the sparkline and the number of items
19 # has exceeded max_items, any excess values are removed from the left of the
20 # graph, and new values are added to the right.
21
22
23 # The following is an example that shows the
24
25 # setup display
26 # instance sparklines
27 # add to the display
28 # Loop the following steps:
29 #     add new values to sparkline `add_value`
30 #     update the sparklines `update`
31
32 import random

```

(continues on next page)

(continued from previous page)

```
33 import time
34 import board
35 import displayio
36 import terminalio
37 from adafruit_display_text import label
38 from adafruit_display_shapes.sparkline import Sparkline
39
40
41 if "DISPLAY" not in dir(board):
42     # Setup the LCD display with driver
43     # You may need to change this to match the display driver for the chipset
44     # used on your display
45     from adafruit_ili9341 import ILI9341
46
47     displayio.release_displays()
48
49     # setup the SPI bus
50     spi = board.SPI()
51     tft_cs = board.D9 # arbitrary, pin not used
52     tft_dc = board.D10
53     tft_backlight = board.D12
54     tft_reset = board.D11
55
56     while not spi.try_lock():
57         spi.configure(baudrate=32000000)
58     spi.unlock()
59
60     display_bus = displayio.FourWire(
61         spi,
62         command=tft_dc,
63         chip_select=tft_cs,
64         reset=tft_reset,
65         baudrate=32000000,
66         polarity=1,
67         phase=1,
68     )
69
70     print("spi.frequency: {}".format(spi.frequency))
71
72     # Number of pixels in the display
73     DISPLAY_WIDTH = 320
74     DISPLAY_HEIGHT = 240
75
76     # create the display
77     display = ILI9341(
78         display_bus,
79         width=DISPLAY_WIDTH,
80         height=DISPLAY_HEIGHT,
81         rotation=180, # The rotation can be adjusted to match your configuration.
82         auto_refresh=True,
83         native_frames_per_second=90,
84     )
85
86     # reset the display to show nothing.
87     display.show(None)
88 else:
89     # built-in display
```

(continues on next page)

(continued from previous page)

```

90     display = board.DISPLAY
91     DISPLAY_WIDTH = board.DISPLAY.width
92
93     #####
94     # Create background bitmaps and sparklines
95     #####
96
97     # Baseline size of the sparkline chart, in pixels.
98     chart_width = 50
99     chart_height = 50
100
101     font = terminalio.FONT
102
103     # Setup the first bitmap and sparkline
104     # This sparkline has no background bitmap
105     # sparkline1 uses a vertical y range between -1 to +1.25 and will contain a maximum_
106     ↪ of 40 items
107     sparkline1 = Sparkline(
108         width=chart_width,
109         height=chart_height,
110         max_items=40,
111         y_min=-1,
112         y_max=1.25,
113         x=10,
114         y=10,
115     )
116
117     # Label the y-axis range
118     text_labella = label.Label(
119         font=font, text=str(sparkline1.y_top), color=0xFFFFFF
120     ) # y_top label
121     text_labella.anchor_point = (0, 0.5) # set the anchorpoint
122     text_labella.anchored_position = (
123         10 + chart_width,
124         10,
125     ) # set the text anchored position to the upper right of the graph
126
127     text_labellb = label.Label(
128         font=font, text=str(sparkline1.y_bottom), color=0xFFFFFF
129     ) # y_bottom label
130     text_labellb.anchor_point = (0, 0.5) # set the anchorpoint
131     text_labellb.anchored_position = (
132         10 + chart_width,
133         10 + chart_height,
134     ) # set the text anchored position to the upper right of the graph
135
136     # Setup the second bitmap and sparkline
137     # sparkline2 uses a vertical y range between 0 to 1, and will contain a
138     # maximum of 10 items
139     #
140     palette2 = displayio.Palette(1) # color palette used for bitmap2 (one color)
141     palette2[0] = 0x0000FF
142
143     bitmap2 = displayio.Bitmap(chart_width * 2, chart_height * 2, 1) # create bitmap2
144     tilegrid2 = displayio.TileGrid(
145         bitmap2, pixel_shader=palette2, x=150, y=10

```

(continues on next page)

(continued from previous page)

```

146 ) # Add bitmap2 to tilegrid2
147 sparkline2 = Sparkline(
148     width=chart_width * 2,
149     height=chart_height * 2,
150     max_items=10,
151     y_min=0,
152     y_max=1,
153     x=150,
154     y=10,
155     color=0xFF00FF,
156 )
157
158
159 # Setup the third bitmap and third sparkline
160 # sparkline3 contains a maximum of 10 items
161 # since y_min and y_max are not specified, sparkline3 uses autoranging for both
162 # the top and bottom of the y-axis.
163 # Note1: Any unspecified edge limit (y_min or y_max) will autorange that edge based
164 # on the data in the list.
165 # Note2: You can read back the current value of the y-axis limits by using
166 # sparkline3.y_bottom or sparkline3.y_top
167
168
169 palette3 = displayio.Palette(1) # color palette used for bitmap (one color)
170 palette3[0] = 0x11FF44
171 bitmap3 = displayio.Bitmap(DISPLAY_WIDTH - 30, chart_height * 2, 1) # create bitmap3
172 tilegrid3 = displayio.TileGrid(
173     bitmap3, pixel_shader=palette3, x=0, y=120
174 ) # Add bitmap3 to tilegrid3
175
176 sparkline3 = Sparkline(
177     width=DISPLAY_WIDTH - 30,
178     height=chart_height * 2,
179     max_items=10,
180     x=0,
181     y=120,
182     color=0xFFFFFF,
183 )
184
185 # Initialize the y-axis labels for mySparkline3 with no text
186 text_label3a = label.Label(font=font, text="", color=0x11FF44) # y_top label
187 text_label3a.anchor_point = (0, 0.5) # set the anchorpoint
188 text_label3a.anchored_position = (
189     sparkline3.width,
190     120,
191 ) # set the text anchored position to the upper right of the graph
192
193 text_label3b = label.Label(font=font, text="", color=0x11FF44) # y_bottom label
194 text_label3b.anchor_point = (0, 0.5) # set the anchorpoint
195 text_label3b.anchored_position = (
196     sparkline3.width,
197     120 + sparkline3.height,
198 ) # set the text anchored position to the upper right of the graph
199
200 # Create a group to hold the three bitmap TileGrids and the three sparklines and
201 # append them into the group (my_group)
202 #

```

(continues on next page)

(continued from previous page)

```
203 # Note: In cases where display elements will overlap, then the order the elements
204 # are added to the group will set which is on top. Latter elements are displayed
205 # on top of former elements.
206 my_group = displayio.Group()
207
208 my_group.append(sparkline1)
209 my_group.append(text_label1a)
210 my_group.append(text_label1b)
211
212 my_group.append(tilegrid2)
213 my_group.append(sparkline2)
214
215 my_group.append(tilegrid3)
216 my_group.append(sparkline3)
217 my_group.append(text_label3a)
218 my_group.append(text_label3b)
219
220 # Set the display to show my_group that contains all the bitmap TileGrids and
221 # sparklines
222 display.show(my_group)
223
224 i = 0 # This is a counter for changing the random values for mySparkline3
225
226 # Start the main loop
227 while True:
228
229     # Turn off auto_refresh to prevent partial updates of the screen during updates
230     # of the sparklines
231     display.auto_refresh = False
232
233     # add_value: add a new value to a sparkline
234     # Note: The y-range for sparkline1 is set to -1 to 1.25, so all these random
235     # values (between 0 and 1) will fit within the visible range of this sparkline
236     sparkline1.add_value(random.uniform(0, 1))
237
238     # Note: For sparkline2, the y-axis range is set from 0 to 1.
239     # With the random values set between -1 and +2, the values will sometimes
240     # be out of the y-range. This example shows how the fixed y-range (0 to 1)
241     # will "clip" values (it will not display them) that are above or below the
242     # y-range.
243     sparkline2.add_value(random.uniform(-1, 2))
244
245     # sparkline3 is set autoranging for both the top and bottom of the Y-axis
246
247     # In this example, for 15 values, this adds points in the range from 0 to 1.
248     # Then, for the next 15 values, it adds points in the range of 0 to 10.
249     # This is to highlight the autoranging of the y-axis.
250     # Notice how the y-axis labels show that the y-scale is changing.
251     #
252     # An exercise for the reader: You can set only one or the other sparkline axis
253     # to autoranging by setting its value to None.
254     if i < 15:
255         sparkline3.add_value(random.uniform(0, 1))
256     else:
257         sparkline3.add_value(random.uniform(0, 10))
258     text_label3a.text = str(sparkline3.y_top)
259     text_label3b.text = str(sparkline3.y_bottom)
```

(continues on next page)

(continued from previous page)

```

260     i += 1 # increment the counter
261     if i > 30: # After 30 times through the loop, reset the counter
262         i = 0
263
264     # Turn on auto_refresh for the display
265     display.auto_refresh = True
266
267     # The display seems to be less jittery if a small sleep time is provided
268     # You can adjust this to see if it has any effect
269     time.sleep(0.01)

```

6.6 Circle Animation

Example showing the features of the new Circle setter

Listing 6: examples/display_shapes_circle_animation.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """
5  This is an animation to demonstrate the use of Circle Setter Attribute.
6  """
7
8  import time
9  import gc
10 import board
11 import displayio
12 from adafruit_display_shapes.circle import Circle
13
14 # use built in display (MagTag, PyPortal, PyGamer, PyBadge, CLUE, etc.)
15 # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.
16 # ↪ https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-
17 ↪ and-display-bus)
18 display = board.DISPLAY
19
20 # Make the display context
21 main_group = displayio.Group()
22
23 # Make a background color fill
24 color_bitmap = displayio.Bitmap(display.width, display.height, 1)
25 color_palette = displayio.Palette(1)
26 color_palette[0] = 0xFFFFFF
27 bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
28 main_group.append(bg_sprite)
29
30 # Setting up the Circle starting position
31 posx = 50
32 posy = 50
33
34 # Define Circle characteristics
35 circle_radius = 20
36 circle = Circle(posx, posy, circle_radius, fill=0x00FF00, outline=0xFF00FF)

```

(continues on next page)

(continued from previous page)

```

36 main_group.append(circle)
37
38 # Define Circle Animation Steps
39 delta_x = 2
40 delta_y = 2
41
42 # Showing the items on the screen
43 display.show(main_group)
44
45 while True:
46
47     if circle.y + circle_radius >= display.height - circle_radius:
48         delta_y = -1
49     if circle.x + circle_radius >= display.width - circle_radius:
50         delta_x = -1
51     if circle.x - circle_radius <= 0 - circle_radius:
52         delta_x = 1
53     if circle.y - circle_radius <= 0 - circle_radius:
54         delta_y = 1
55
56     circle.x = circle.x + delta_x
57     circle.y = circle.y + delta_y
58
59     time.sleep(0.02)
60     gc.collect()

```

6.7 circle

Various common shapes for use with displayio - Circle shape!

- Author(s): Limor Fried

6.7.1 Implementation Notes

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>

class adafruit_display_shapes.circle.Circle(x0, y0, r, *, fill=None, outline=None, stroke=1)

A circle.

Parameters

- **x0** – The x-position of the center.
- **y0** – The y-position of the center.
- **r** – The radius of the circle.
- **fill** – The color to fill the circle. Can be a hex value for a color or None for transparent.
- **outline** – The outline of the circle. Can be a hex value for a color or None for no outline.
- **stroke** – Used for the outline. Will not change the radius.

x0

The x-position of the center of the circle.

y0

The y-position of the center of the circle.

6.8 rect

Various common shapes for use with displayio - Rectangle shape!

- Author(s): Limor Fried

6.8.1 Implementation Notes

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>

class `adafruit_display_shapes.rect.Rect` (*x*, *y*, *width*, *height*, *, *fill=None*, *outline=None*, *stroke=1*)

A rectangle.

Parameters

- **x** – The x-position of the top left corner.
- **y** – The y-position of the top left corner.
- **width** – The width of the rectangle.
- **height** – The height of the rectangle.
- **fill** – The color to fill the rectangle. Can be a hex value for a color or `None` for transparent.
- **outline** – The outline of the rectangle. Can be a hex value for a color or `None` for no outline.
- **stroke** – Used for the outline. Will not change the outer bound size set by `width` and `height`.

fill

The fill of the rectangle. Can be a hex value for a color or `None` for transparent.

outline

The outline of the rectangle. Can be a hex value for a color or `None` for no outline.

6.9 roundrect

A slightly modified version of `Adafruit_CircuitPython_Display_Shapes` that includes an explicit call to `palette.make_opaque()` in the fill color setter function.

class `adafruit_display_shapes.roundrect.RoundRect` (*x*, *y*, *width*, *height*, *r*, *, *fill=None*, *outline=None*, *stroke=1*)

A round-corner rectangle.

Parameters

- **x** – The x-position of the top left corner.
- **y** – The y-position of the top left corner.

- **width** – The width of the rounded-corner rectangle.
- **height** – The height of the rounded-corner rectangle.
- **r** – The radius of the rounded corner.
- **fill** – The color to fill the rounded-corner rectangle. Can be a hex value for a color or `None` for transparent.
- **outline** – The outline of the rounded-corner rectangle. Can be a hex value for a color or `None` for no outline.
- **stroke** – Used for the outline. Will not change the outer bound size set by `width` and `height`.

fill

The fill of the rounded-corner rectangle. Can be a hex value for a color or `None` for transparent.

outline

The outline of the rounded-corner rectangle. Can be a hex value for a color or `None` for no outline.

6.10 triangle

Various common shapes for use with displayio - Triangle shape!

- Author(s): Melissa LeBlanc-Williams

6.10.1 Implementation Notes

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>

```
class adafruit_display_shapes.triangle.Triangle (x0, y0, x1, y1, x2, y2, *, fill=None, outline=None)
```

A triangle.

Parameters

- **x0** – The x-position of the first vertex.
- **y0** – The y-position of the first vertex.
- **x1** – The x-position of the second vertex.
- **y1** – The y-position of the second vertex.
- **x2** – The x-position of the third vertex.
- **y2** – The y-position of the third vertex.
- **fill** – The color to fill the triangle. Can be a hex value for a color or `None` for transparent.
- **outline** – The outline of the triangle. Can be a hex value for a color or `None` for no outline.

fill

The fill of the triangle. Can be a hex value for a color or `None` for transparent.

6.11 line

Various common shapes for use with displayio - Line shape!

- Author(s): Melissa LeBlanc-Williams

6.11.1 Implementation Notes

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>

class `adafruit_display_shapes.line.Line(x0, y0, x1, y1, color)`

A line.

Parameters

- **x0** – The x-position of the first vertex.
- **y0** – The y-position of the first vertex.
- **x1** – The x-position of the second vertex.
- **y1** – The y-position of the second vertex.
- **color** – The color of the line.

6.12 polygon

Various common shapes for use with displayio - Polygon shape!

- Author(s): Melissa LeBlanc-Williams

6.12.1 Implementation Notes

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>

class `adafruit_display_shapes.polygon.Polygon(points, *, outline=None)`

A polygon.

Parameters

- **points** – A list of (x, y) tuples of the points
- **outline** – The outline of the polygon. Can be a hex value for a color or None for no outline.

outline

The outline of the polygon. Can be a hex value for a color or None for no outline.

6.13 sparkline

Various common shapes for use with displayio - Sparkline!

- Author(s): Kevin Matocha

6.13.1 Implementation Notes

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>

```
class adafruit_display_shapes.sparkline.Sparkline (width, height, max_items,  
                                                y_min=None, y_max=None, x=0,  
                                                y=0, color=16777215)
```

A sparkline graph.

Parameters

- **width** – Width of the sparkline graph in pixels
- **height** – Height of the sparkline graph in pixels
- **max_items** – Maximum number of values housed in the sparkline
- **y_min** – Lower range for the y-axis. Set to None for autorange.
- **y_max** – Upper range for the y-axis. Set to None for autorange.
- **x** – X-position on the screen, in pixels
- **y** – Y-position on the screen, in pixels
- **color** – Line color, the default value is 0xFFFFFF (WHITE)

add_value (*value*)

Add a value to the sparkline. :param value: The value to be added to the sparkline

clear_values ()

Removes all values from the `_spark_list` list and removes all lines in the group

update ()

Update the drawing of the sparkline.

values ()

Returns the values displayed on the sparkline.

CHAPTER 7

Indices and tables

- `genindex`
- `modindex`
- `search`

a

`adafruit_display_shapes.circle`, 28
`adafruit_display_shapes.line`, 30
`adafruit_display_shapes.polygon`, 31
`adafruit_display_shapes.rect`, 29
`adafruit_display_shapes.roundrect`, 29
`adafruit_display_shapes.sparkline`, 31
`adafruit_display_shapes.triangle`, 30

A

`adafruit_display_shapes.circle` (module), 28
`adafruit_display_shapes.line` (module), 30
`adafruit_display_shapes.polygon` (module), 31
`adafruit_display_shapes.rect` (module), 29
`adafruit_display_shapes.roundrect` (module), 29
`adafruit_display_shapes.sparkline` (module), 31
`adafruit_display_shapes.triangle` (module), 30
`add_value()` (`adafruit_display_shapes.sparkline.Sparkline` method), 32

C

`Circle` (class in `adafruit_display_shapes.circle`), 28
`clear_values()` (`adafruit_display_shapes.sparkline.Sparkline` method), 32

F

`fill` (`adafruit_display_shapes.rect.Rect` attribute), 29
`fill` (`adafruit_display_shapes.roundrect.RoundRect` attribute), 30
`fill` (`adafruit_display_shapes.triangle.Triangle` attribute), 30

L

`Line` (class in `adafruit_display_shapes.line`), 31

O

`outline` (`adafruit_display_shapes.polygon.Polygon` attribute), 31
`outline` (`adafruit_display_shapes.rect.Rect` attribute), 29
`outline` (`adafruit_display_shapes.roundrect.RoundRect` attribute), 30

P

`Polygon` (class in `adafruit_display_shapes.polygon`), 31

R

`Rect` (class in `adafruit_display_shapes.rect`), 29
`RoundRect` (class in `adafruit_display_shapes.roundrect`), 29

S

`Sparkline` (class in `adafruit_display_shapes.sparkline`), 32

T

`Triangle` (class in `adafruit_display_shapes.triangle`), 30

U

`update()` (`adafruit_display_shapes.sparkline.Sparkline` method), 32

V

`values()` (`adafruit_display_shapes.sparkline.Sparkline` method), 32

X

`x0` (`adafruit_display_shapes.circle.Circle` attribute), 28

Y

`y0` (`adafruit_display_shapes.circle.Circle` attribute), 29