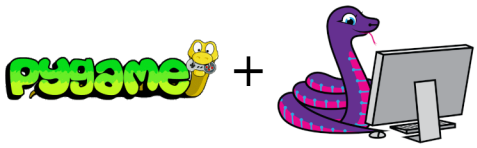
Blinka_Displayio_PyGameDisplay Library Documen

Release 1.0

Tim C

Feb 13, 2023

CONTENTS



Blinka makes her debut on the big screen! With this library you can use CircuitPython `displayio` code on PC and Raspberry Pi to output to a PyGame window instead of a hardware display connected to I2C or SPI. This makes it easy to to use `displayio` elements on HDMI and other large format screens.

Warning: you must check `display.running` in the main loop to correctly handle the close button!

CHAPTER

ONE

DEPENDENCIES

This driver depends on:

- PyGame
- Adafruit Blinka Displayio

Please ensure all dependencies are available they can be installed with pip3

CHAPTER

TWO

OPTIONAL DEPENDENCIES

This driver can optionally make use of these `displayio` module libraries:

- Adafruit Display Text
- Adafruit ImageLoad
- Adafruit Progress Bar
- Adafruit Display Button

They can be installed with pip3.

CHAPTER

THREE

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 blinka-displayio-pygamedisplay
```

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

```
sudo pip3 install blinka-displayio-pygamedisplay
```

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 blinka-displayio-pygamedisplay
```

CHAPTER

FOUR

USAGE EXAMPLE

```
import displayio
from blinka_displayio_pygamedisplay import PyGameDisplay

display = PyGameDisplay(width=320, height=240)
splash = displayio.Group()
display.show(splash)

color_bitmap = displayio.Bitmap(display.width, display.height, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x00FF00  # Bright Green

bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)
# Must check display.running in the main loop!

while True:
    if display.check_quit():
        break
```

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER

SIX

DOCUMENTATION

For information on building library documentation, please check out this guide.

CHAPTER

SEVEN

TABLE OF CONTENTS

7.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/blinka_displayio_pygamedisplay_simpletest.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
Make green and purple rectangles and a
"Hello World" label.
"""
import time

import displayio
import rainbowio
import terminalio
from adafruit_display_text import label
from blinka_displayio_pygamedisplay import PyGameDisplay


# Make the display context
display = PyGameDisplay(icon="blinka.png", width=400, height=300)

# Make the display context
splash = displayio.Group()
display.show(splash)

# Draw a green background
color_bitmap = displayio.Bitmap(display.width, display.height, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x00FF00  # Bright Green

bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)

splash.append(bg_sprite)

# Draw a smaller inner rectangle
inner_bitmap = displayio.Bitmap(display.width - 40, display.height - 40, 1)
```

(continues on next page)

(continued from previous page)

```
inner_palette = displayio.Palette(1)
inner_palette[0] = 0xAA0088  # Purple
inner_sprite = displayio.TileGrid(inner_bitmap, pixel_shader=inner_palette, x=20, y=20)
splash.append(inner_sprite)

# Draw a label
text_group = displayio.Group()
text_area = label.Label(terminalio.FONT, text="Hello World!", color=0xFFFF00, scale=4)
text_area.anchor_point = (0.5, 0.5)
text_area.anchored_position = (display.width // 2, display.height // 2)

# text_group.append(text_area)  # Subgroup for text scaling
splash.append(text_area)

color_num = 0
while True:
    text_area.color = rainbowio.colorwheel(color_num)
    color_num += 1
    if color_num > 255:
        color_num = 0
    print(time.monotonic())
    time.sleep(0.05)

    if display.check_quit():
        break
```

7.2 Anchor test

Testing anchor point of adafruit_display_text label.

Listing 2: examples/blinka_displayio_pygamedisplay_anchor_test.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
Testing anchor point of adafruit_display_text label.
"""
import terminalio
import displayio
from adafruit_display_text import label
from blinka_displayio_pygamedisplay import PyGameDisplay

display = PyGameDisplay(width=1770, height=920)

text_area = label.Label(terminalio.FONT, text="Hello world", scale=3)

text_area.anchor_point = (0.5, 0.5)
text_area.anchored_position = (display.width // 2, display.height // 2)
print(text_area.bounding_box)
```

(continues on next page)

(continued from previous page)

```
print(f"{text_area.x}, {text_area.y}")
main_group = displayio.Group()
main_group.append(text_area)
display.show(main_group)

# text_area.y = 37
while True:
    if display.check_quit():
        break
```

7.3 Button example

Initialize the PyGame display and add a button to it. React to click events on the button.

Listing 3: examples/blinka_displayio_pygamedisplay_button_example.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
Initialize the PyGame display and add a button to it.
React to click events on the button
"""
import displayio
import pygame
import terminalio
from adafruit_button import Button
from blinka_displayio_pygamedisplay import PyGameDisplay

# --| Button Config |-----------------------------------------------
BUTTON_X = 110
BUTTON_Y = 95
BUTTON_WIDTH = 100
BUTTON_HEIGHT = 50
BUTTON_STYLE = Button.ROUNDRECT
BUTTON_FILL_COLOR = 0x00FFFF
BUTTON_OUTLINE_COLOR = 0xFF00FF
BUTTON_LABEL = "HELLO WORLD"
BUTTON_LABEL_COLOR = 0x000000
# --| Button Config |-----------------------------------------------

display = PyGameDisplay(width=320, height=240)
splash = displayio.Group()
display.show(splash)

GREEN = 0x00FF00
BLUE = 0x0000FF
CUR_COLOR = GREEN

color_bitmap = displayio.Bitmap(display.width, display.height, 1)
```

(continues on next page)

(continued from previous page)

```
color_palette = displayio.Palette(1)
color_palette[0] = CUR_COLOR  # Bright Green
print(color_palette[0])
# Make the button
button = Button(
    x=BUTTON_X,
    y=BUTTON_Y,
    width=BUTTON_WIDTH,
    height=BUTTON_HEIGHT,
    style=BUTTON_STYLE,
    fill_color=BUTTON_FILL_COLOR,
    outline_color=BUTTON_OUTLINE_COLOR,
    label="HELLO WORLD",
    label_font=terminalio.FONT,
    label_color=BUTTON_LABEL_COLOR,
)

button.width = 130

bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)

splash.append(button)

button.body.fill = 0x0000FF
# pylint: disable=no-member

# Must check display.running in the main loop!
while True:
    # get mouse up  events
    ev = pygame.event.get(eventtype=pygame.MOUSEBUTTONUP)
    # proceed events
    for event in ev:
        pos = pygame.mouse.get_pos()
        print(pos)
        button.selected = False
        if button.contains(pos):
            if CUR_COLOR == GREEN:
                print("change to blue")
                color_palette[0] = BLUE
                CUR_COLOR = BLUE
            else:
                color_palette[0] = GREEN
                CUR_COLOR = GREEN
    # get mouse down  events
    ev = pygame.event.get(eventtype=pygame.MOUSEBUTTONDOWN)
    for event in ev:
        pos = pygame.mouse.get_pos()
        print(pos)
        if button.contains(pos):
            button.selected = True

```

(continues on next page)

(continued from previous page)

```
    if display.check_quit():
        break
```

7.4 Imageload BMP

Use adafruit_imageload to show a bitmap on the screen

Listing 4: examples/blinka_displayio_pygamedisplay_imageload_bmp_test.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
Use adafruit_imageload to show a bitmap on the screen
"""
import displayio
import adafruit_imageload
from blinka_displayio_pygamedisplay import PyGameDisplay

display = PyGameDisplay(icon="blinka.png", width=800, height=600)

bitmap, palette = adafruit_imageload.load(
    "robot_friend.bmp", bitmap=displayio.Bitmap, palette=displayio.Palette
)

# Create a TileGrid to hold the bitmap
tile_grid = displayio.TileGrid(bitmap, pixel_shader=palette)

# Create a Group to hold the TileGrid
img_group = displayio.Group()

# Add the TileGrid to the Group
img_group.append(tile_grid)

# Add the Group to the Display
display.show(img_group)

# Loop forever so you can enjoy your image
while True:
    if display.check_quit():
        break
```

7.5 PartyParrot

Party parrot animation code adapted from: https://github.com/adafruit/Adafruit_Learning_System_Guides/tree/master/IoT_Party_Parrot

Listing 5: examples/blinka_displayio_pygamedisplay_partyparrot.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
Party parrot animation code adapted from:
https://github.com/adafruit/Adafruit_Learning_System_Guides/tree/master/IoT_Party_Parrot

Thank you @BlitzCityDIY
"""
import time
import adafruit_imageload
import displayio
from blinka_displayio_pygamedisplay import PyGameDisplay

display = PyGameDisplay(width=320, height=320)

group = displayio.Group(scale=10)

#  get the spritesheet from here:
#  https://github.com/adafruit/Adafruit_Learning_System_Guides/tree/master/IoT_Party_
↪Parrot

#  load in party parrot bitmap
parrot_bit, parrot_pal = adafruit_imageload.load(
    "partyParrotsTweet.bmp", bitmap=displayio.Bitmap, palette=displayio.Palette
)

parrot_grid = displayio.TileGrid(
    parrot_bit,
    pixel_shader=parrot_pal,
    width=1,
    height=1,
    tile_height=32,
    tile_width=32,
    default_tile=10,
    x=0,
    y=0,
)

group.append(parrot_grid)

display.show(group)

parrot = True  #  state to track if an animation is currently running
party = 0  #  time.monotonic() holder
p = 0  #  index for tilegrid
```

(continues on next page)

(continued from previous page)

```
party_count = 0  #  count for animation cycles

while True:
    #  when a new tweet comes in...
    if parrot:
        #  every 0.1 seconds...
        if (party + 0.1) < time.monotonic():
            #  the party parrot animation cycles
            parrot_grid[0] = p
            #  p is the tilegrid index location
            p += 1
            party = time.monotonic()
            #  if an animation cycle ends
            if p > 9:
                #  index is reset
                p = 0
                #  animation cycle count is updated
                party_count += 1
                print("party parrot", party_count)

    if display.check_quit():
        break
```

7.6 Progressbar

Example showing the use of adafruit_progressbar

Listing 6: examples/blinka_displayio_pygamedisplay_progressbar_example.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
example showing the use of adafruit_progressbar
"""
import time
import displayio
from adafruit_progressbar.adafruit_progressbar import ProgressBar
from blinka_displayio_pygamedisplay import PyGameDisplay

# Make the display context
splash = displayio.Group(scale=2)

display = PyGameDisplay(width=480, height=320)

color_bitmap = displayio.Bitmap(display.width, display.height, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x0000FF

bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
```

(continues on next page)

(continued from previous page)

```
splash.append(bg_sprite)

display.show(splash)

# set progress bar width and height relative to board's display
width = display.width - 40
height = 30

x = display.width // 2 - width // 2
y = display.height // 3

# Create a new progress_bar object at (x, y)
progress_bar = ProgressBar(x, y, width, height, 1.0)

# Append progress_bar to the splash group
splash.append(progress_bar)

current_progress = 0.0
while True:
    # range end is exclusive so we need to use 1 bigger than max number that we want
    for current_progress in range(0, 101, 1):
        print("Progress: {}%".format(current_progress))
        progress_bar.progress = current_progress / 100  # convert to decimal
        time.sleep(0.01)
    time.sleep(0.3)
    progress_bar.progress = 0.0
    time.sleep(0.3)

    if display.check_quit():
        break
```

7.7 Pyportal Bitcoin

This example will access the coindesk API, grab a number like bitcoin value in USD and display it on a screen If you can find something that spits out JSON data, we can display it! You can find any resources in the associated Learn Guide at: https://learn.adafruit.com/pyportal-bitcoin-value-display

Listing 7: examples/blinka_displayio_pygamedisplay_pyportal_bitcoin.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
This example will access the coindesk API, grab a number like bitcoin value in
USD and display it on a screen
If you can find something that spits out JSON data, we can display it!
You can find any resources in the associated Learn Guide at:
https://learn.adafruit.com/pyportal-bitcoin-value-display
"""
import os
```

(continues on next page)

(continued from previous page)

```
import time
from adafruit_pyportal import PyPortal
from secret_credentials import secrets
from blinka_displayio_pygamedisplay import PyGameDisplay

# Make the display context
display = PyGameDisplay(icon="blinka.png", width=320, height=240)
# You can display in 'GBP', 'EUR' or 'USD'
CURRENCY = "USD"
# Set up where we'll be fetching data from
DATA_SOURCE = "https://api.coindesk.com/v1/bpi/currentprice.json"
DATA_LOCATION = ["bpi", CURRENCY, "rate_float"]


def text_transform(val):
    """Format value with currency symbol"""
    if CURRENCY == "USD":
        return "$%d" % val
    if CURRENCY == "EUR":
        return "€%d" % val
    if CURRENCY == "GBP":
        return "£%d" % val
    return "%d" % val


# the current working directory (where this file is)
try:
    cwd = os.path.dirname(os.path.realpath(__file__))
except AttributeError:
    cwd = ("/" + __file__).rsplit("/", 1)[0]

pyportal = PyPortal(
    external_spi="fake",
    url=DATA_SOURCE,
    json_path=DATA_LOCATION,
    default_bg=cwd + "/bitcoin_background.bmp",
    text_font=cwd + "/fonts/Arial-Bold-24-Complete.bdf",
    text_position=(195, 130),
    text_color=0x0,
    text_transform=text_transform,
    display=display,
    secrets=secrets,
)
pyportal.preload_font(b"$012345789")  # preload numbers
pyportal.preload_font((0x00A3, 0x20AC))  # preload gbp/euro symbol

while True:
    try:
        value = pyportal.fetch()
        print("Response is", value)
    except (ValueError, RuntimeError) as e:
        print("Some error occured, retrying! -", e)
```

(continues on next page)

(continued from previous page)

```
    time.sleep(3 * 60)  # wait 3 minutes

    if display.check_quit():
        break
```

7.8 Readme Example

Initialize the PyGame display and fill it with green

Listing 8: examples/blinka_displayio_pygamedisplay_readme_example.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
Initialize the PyGame display and fill it with green
"""
import displayio
from blinka_displayio_pygamedisplay import PyGameDisplay

display = PyGameDisplay(width=320, height=240)
splash = displayio.Group()
display.show(splash)

color_bitmap = displayio.Bitmap(display.width, display.height, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0x00FF00  # Bright Green

bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)


while True:
    if display.check_quit():
        break
```

7.9 SETUP Only

Make green and purple rectangles and a "Hello World" label.

Listing 9: examples/blinka_displayio_pygamedisplay_setup_only.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
Make green and purple rectangles and a
```

(continues on next page)

(continued from previous page)

```
"Hello World" label.
"""
import displayio
from blinka_displayio_pygamedisplay import PyGameDisplay


# Make the display context. Change size if you want
display = PyGameDisplay(width=320, height=240)

# Make the display context
main_group = displayio.Group()
display.show(main_group)


while True:
    if display.check_quit():
        break
```

7.10 Shapes

This is adapted from an example in the shapes library to work with pygame display. It shows how to draw various different shapes and place them on the screen

Listing 10: examples/blinka_displayio_pygamedisplay_shapes.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense

"""
This is adapted from an example in the shapes library to work with pygame display.
It shows how to draw various different shapes and place them on the screen.
"""

import displayio
from adafruit_display_shapes.rect import Rect
from adafruit_display_shapes.circle import Circle
from adafruit_display_shapes.roundrect import RoundRect
from adafruit_display_shapes.triangle import Triangle
from adafruit_display_shapes.line import Line
from adafruit_display_shapes.polygon import Polygon
from blinka_displayio_pygamedisplay import PyGameDisplay

# Make the display context
splash = displayio.Group(scale=2)

display = PyGameDisplay(icon="blinka.png", width=640, height=480)
display.show(splash)

# Make a background color fill
```

(continues on next page)

(continued from previous page)

```
color_bitmap = displayio.Bitmap(320, 240, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0xFFFFFF
bg_sprite = displayio.TileGrid(color_bitmap, x=0, y=0, pixel_shader=color_palette)
splash.append(bg_sprite)
################################################################################

splash.append(Line(220, 130, 270, 210, 0xFF0000))
splash.append(Line(270, 210, 220, 210, 0xFF0000))
splash.append(Line(220, 210, 270, 130, 0xFF0000))
splash.append(Line(270, 130, 220, 130, 0xFF0000))

# Draw a blue star
polygon = Polygon(
    [
        (255, 40),
        (262, 62),
        (285, 62),
        (265, 76),
        (275, 100),
        (255, 84),
        (235, 100),
        (245, 76),
        (225, 62),
        (248, 62),
    ],
    outline=0x0000FF,
)
polygon.x += 150
polygon.y += 50
splash.append(polygon)

triangle = Triangle(170, 50, 120, 140, 210, 160, fill=0x00FF00, outline=0xFF00FF)
triangle.x += 240
triangle.y += 180
splash.append(triangle)

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

circle = Circle(100, 30, 20, fill=0x00FF00, outline=0xFF00FF)
circle.x += 200
splash.append(circle)

print(circle.fill)

rect2 = Rect(50, 100, 61, 81, outline=0x0, stroke=3)
rect2.y += 10
splash.append(rect2)


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

(continues on next page)

(continued from previous page)

```
roundrect.y += 270
splash.append(roundrect)


while True:
    if display.check_quit():
        break
```

7.11 Shapes Sparkline

This example has been adapted from the example in the shapes library to work with pygame display.

Listing 11: examples/blinka_displayio_pygamedisplay_shapes_sparkline.py

```
# SPDX-FileCopyrightText: 2020 Kevin Matocha, Tim C
#
# SPDX-License-Identifier: Unlicense
"""
This example has been adapted from the example in the shapes
library to work with pygame display.
"""

# class of sparklines in CircuitPython
# created by Kevin Matocha - Copyright 2020 (C)

# See the bottom for a code example using the `sparkline` Class.

# # File: display_shapes_sparkline.py
# A sparkline is a scrolling line graph, where any values added to sparkline
# using `add_value` are plotted.
#
# The `sparkline` class creates an element suitable for adding to the display
# using `display.show(mySparkline)` or adding to a `displayio.Group` to be displayed.
#
# When creating the sparkline, identify the number of `max_items` that will be
# included in the graph.
# When additional elements are added to the sparkline and the number of items
# has exceeded max_items, any excess values are removed from the left of the
# graph, and new values are added to the right.


# The following is an example that shows the

# setup display
# instance sparklines
# add to the display
# Loop the following steps:
#         add new values to sparkline `add_value`
#         update the sparklines `update`

```

(continues on next page)

(continued from previous page)

```
import random
import time
import displayio
import terminalio
from adafruit_display_text import label
from adafruit_display_shapes.sparkline import Sparkline
from adafruit_display_shapes.line import Line
from adafruit_display_shapes.rect import Rect
from blinka_displayio_pygamedisplay import PyGameDisplay


########################################
# Create background bitmaps and sparklines
########################################

display = PyGameDisplay(icon="blinka.png", width=640, height=480)

# Baseline size of the sparkline chart, in pixels.
chart_width = display.width - 50
chart_height = display.height - 50

font = terminalio.FONT

LINE_COLOR = 0xFFFFFF

# Setup the first bitmap and sparkline
# This sparkline has no background bitmap
# mySparkline1 uses a vertical y range between 0 to 10 and will contain a
# maximum of 40 items
sparkline1 = Sparkline(
    width=chart_width,
    height=chart_height,
    max_items=40,
    y_min=0,
    y_max=10,
    x=40,
    y=30,
    color=LINE_COLOR,
)

# Label the y-axis range

TEXT_XOFFSET = -10
text_label1a = label.Label(
    font=font, text=str(sparkline1.y_top), color=LINE_COLOR
)  # yTop label
text_label1a.anchor_point = (1, 0.5)  # set the anchorpoint at right-center
text_label1a.anchored_position = (
    sparkline1.x + TEXT_XOFFSET,
    sparkline1.y,
)  # set the text anchored position to the upper right of the graph
```

(continues on next page)

(continued from previous page)

```
text_label1b = label.Label(
    font=font, text=str(sparkline1.y_bottom), color=LINE_COLOR
)  # yTop label
text_label1b.anchor_point = (1, 0.5)  # set the anchorpoint at right-center
text_label1b.anchored_position = (
    sparkline1.x + TEXT_XOFFSET,
    sparkline1.y + chart_height,
)  # set the text anchored position to the upper right of the graph


bounding_rectangle = Rect(
    sparkline1.x, sparkline1.y, chart_width, chart_height, outline=LINE_COLOR
)


# Create a group to hold the sparkline, text, rectangle and tickmarks
# append them into the group (my_group)
#
# Note: In cases where display elements will overlap, then the order the
# elements are added to the group will set which is on top.  Latter elements
# are displayed on top of former elemtns.

my_group = displayio.Group()

my_group.append(sparkline1)
my_group.append(text_label1a)
my_group.append(text_label1b)
my_group.append(bounding_rectangle)

TOTAL_TICKS = 10

for i in range(TOTAL_TICKS + 1):
    x_start = sparkline1.x - 5
    x_end = sparkline1.x
    y_both = int(round(sparkline1.y + (i * (chart_height) / (TOTAL_TICKS))))
    if y_both > sparkline1.y + chart_height - 1:
        y_both = sparkline1.y + chart_height - 1
    my_group.append(Line(x_start, y_both, x_end, y_both, color=LINE_COLOR))


# Set the display to show my_group that contains the sparkline and other graphics
display.show(my_group)

# Start the main loop
while True:

    # Turn off auto_refresh to prevent partial updates of the screen during updates
    # of the sparkline drawing
    # display.auto_refresh = False

    # add_value: add a new value to a sparkline
```

(continues on next page)

(continued from previous page)

```
    # Note: The y-range for mySparkline1 is set to 0 to 10, so all these random
    # values (between 0 and 10) will fit within the visible range of this sparkline
    sparkline1.add_value(random.uniform(0, 10))

    # Turn on auto_refresh for the display
    # display.auto_refresh = True

    # The display seems to be less jittery if a small sleep time is provided
    # You can adjust this to see if it has any effect
    time.sleep(0.01)

    if display.check_quit():
        break
```

7.12 BitmapLabel Ascent and Descent Test

Make green and purple rectangles and a "Hello World" label.

Listing 12: examples/display_text_bitmap_label_ascent_descent_test.py

```
# SPDX-FileCopyrightText: 2020 Tim C
#
# SPDX-License-Identifier: Unlicense
"""
Make green and purple rectangles and a
"Hello World" label.
"""
import displayio

from adafruit_bitmap_font import bitmap_font

from adafruit_display_text import bitmap_label, label
from blinka_displayio_pygamedisplay import PyGameDisplay

# Make the display context. Change size if you want
display = PyGameDisplay(width=320, height=240)

font = bitmap_font.load_font("font/forkawesome-36.pcf")
w, h, dx, dy = font.get_bounding_box()

glyphs = "".join(chr(0xF000 + i) for i in range(8))

group = displayio.Group()

label = bitmap_label.Label(
    font=font, text=glyphs, background_color=0x0000DD, background_tight=True
)
# label = label.Label(font=font, text=glyphs, background_color=0x0000DD, background_
↪tight=True)

```

(continues on next page)

(continued from previous page)

```
label.anchor_point = (0, 0)
label.anchored_position = (0, 20)

group.append(label)
display.show(group)


while True:
    if display.check_quit():
        break
```

7.13 blinka_displayio_pygamedisplay

Use CircuitPython displayio code on PC and Raspberry Pi output to a PyGame window instead of a physical display.

- Author(s): Tim C

7.13.1 Implementation Notes

Software and Dependencies:

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

class blinka_displayio_pygamedisplay.**PyGameDisplay**(*width=0*, *height=0*, *icon=None*, *caption='Blinka Displayio PyGame'*, *native_frames_per_second=60*, *flags=0*, ***kwargs*)

PyGame display driver

Represents one PyGame window. Uses None for all display hardware parameters.

width - width of the window. A value of zero maximizes the window height - height of the window. A value of zero maximizes the window icon - optional icon for the PyGame window caption - caption for the PyGame window native_frames_per_second - high values result in high cpu-load flags - pygame display-flags, e.g. pygame.FULLSCREEN or pygame.NOFRAME

property auto_refresh: bool

True when the display is refreshed automatically.

check_quit()

Check if the quit button on the window is being pressed.

event_loop(*interval=None*, *on_time=None*, *on_event=None*, *events=None*)

pygame event-loop. Has to be called by the main thread. This method terminates in case of a QUIT-event. An optional callback on_time is executed every interval seconds. Use this callback for application specific logic.

refresh(***, *target_frames_per_second=60*, *minimum_frames_per_second=1*)

While normal display-objects call this method also within a refresh loop, this implementation uses this method only for explicit updates. Note that we cannot just call the update-logic directly, since the pygame-display was created on another thread.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX