
LED*AnimationLibraryDocumentation*

Release 1.0

Roy Hooper

May 18, 2020

Contents

1	Dependencies	3
2	Installing from PyPI	5
3	Usage Example	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	adafruit_led_animation.animation	14
6.2.1	Implementation Notes	14
6.3	adafruit_led_animation.helper	15
6.3.1	Implementation Notes	15
6.4	adafruit_led_animation.group	18
6.4.1	Implementation Notes	18
6.5	adafruit_led_animation.sequence	19
6.5.1	Implementation Notes	19
6.6	adafruit_led_animation.animation.blink	21
6.6.1	Implementation Notes	21
6.7	adafruit_led_animation.animation.solid	21
6.7.1	Implementation Notes	21
6.8	adafruit_led_animation.animation.colorcycle	22
6.8.1	Implementation Notes	22
6.9	adafruit_led_animation.animation.chase	22
6.9.1	Implementation Notes	23
6.10	adafruit_led_animation.animation.comet	23
6.10.1	Implementation Notes	24
6.11	adafruit_led_animation.animation.pulse	24
6.11.1	Implementation Notes	24
6.12	adafruit_led_animation.animation.rainbow	25
6.12.1	Implementation Notes	25
6.13	adafruit_led_animation.animation.sparkle	26

6.13.1	Implementation Notes	26
6.14	<code>adafruit_led_animation.animation.rainbowchase</code>	26
6.14.1	Implementation Notes	26
6.15	<code>adafruit_led_animation.animation.rainbowcomet</code>	27
6.15.1	Implementation Notes	27
6.16	<code>adafruit_led_animation.animation.rainbowsparkle</code>	28
6.16.1	Implementation Notes	28
6.17	<code>adafruit_led_animation.animation.sparklepulse</code>	29
6.17.1	Implementation Notes	29
7	Indices and tables	31
	Python Module Index	33
	Index	35

Perform a variety of LED animation tasks

CHAPTER 1

Dependencies

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](#).

CHAPTER 2

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-led animation
```

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

```
sudo pip3 install adafruit-circuitpython-led animation
```

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-led animation
```


CHAPTER 3

Usage Example

```
import board
import neopixel
from adafruit_led_animation.animation import Blink
import adafruit_led_animation.color as color

# Works on Circuit Playground Express and Bluefruit.
# For other boards, change board.NEOPIXEL to match the pin to which the NeoPixels are
# attached.
pixel_pin = board.NEOPIXEL
# Change to match the number of pixels you have attached to your board.
num_pixels = 10

pixels = neopixel.NeoPixel(pixel_pin, num_pixels)
blink = Blink(pixels, 0.5, color.PURPLE)

while True:
    blink.animate()
```


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 circuitpython-led_animation --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/led_animation_simpletest.py

```
1  """
2  This simpletest example displays the Blink animation.
3
4  For NeoPixel FeatherWing. Update pixel_pin and pixel_num to match your wiring if using
5  a different form of NeoPixels.
6  """
7  import board
8  import neopixel
9  from adafruit_led_animation.animation.blink import Blink
10 from adafruit_led_animation.color import RED
11
12 # Update to match the pin connected to your NeoPixels
13 pixel_pin = board.D6
14 # Update to match the number of NeoPixels you have connected
15 pixel_num = 32
16
17 pixels = neopixel.NeoPixel(pixel_pin, pixel_num, brightness=0.2, auto_write=False)
18
19 blink = Blink(pixels, speed=0.5, color=RED)
20
21 while True:
22     blink.animate()
```

6.2 adafruit_led_animation.animation

Animation base class for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.2.1 Implementation Notes

Hardware:

- [Adafruit NeoPixels](#)
- [Adafruit DotStars](#)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.Animation (pixel_object,      speed,      color,  
                                                  peers=None,      paused=False,  
                                                  name=None)
```

Base class for animations.

add_cycle_complete_receiver (callback)

Adds an additional callback when the cycle completes.

Parameters **callback** – Additional callback to trigger when a cycle completes. The callback is passed the animation object instance.

animate ()

Call `animate()` from your code's main loop. It will draw the animation `draw()` at intervals configured by the `speed` property (set from `init`).

Returns True if the animation draw cycle was triggered, otherwise False.

color

The current color.

cycle_complete ()

Called by some animations when they complete an animation cycle. Animations that support cycle complete notifications will have `X` property set to False. Override as needed.

cycle_count = None

Number of animation cycles completed.

draw ()

Animation subclasses must implement `draw()` to render the animation sequence. Draw must call `show()`.

draw_count = None

Number of animation frames drawn.

fill (color)

Fills the pixel object with a color.

freeze ()

Stops the animation until resumed.

notify_cycles = None

Number of cycles to trigger additional `cycle_done` notifications after

peers = None

A sequence of animations to trigger `.draw()` on when this animation draws.

reset ()
Resets the animation sequence.

resume ()
Resumes the animation.

show ()
Displays the updated pixels. Called during animates with changes.

speed
The animation speed in fractional seconds.

Color variables made available for import for CircuitPython LED animations library.

RAINBOW is a list of colors to use for cycling through.

`adafruit_led_animation.color.colorwheel (pos)`

Input a value 0 to 255 to get a color value. The colours are a transition r - g - b - back to r.

6.3 adafruit_led_animation.helper

Helper classes for making complex animations using CircuitPython LED animations library.

- Author(s): Roy Hooper, Kattni Rembor

6.3.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

class `adafruit_led_animation.helper.PixelMap (strip, pixel_ranges, individual_pixels=False)`

PixelMap lets you treat ranges of pixels as single pixels for animation purposes.

Parameters

- **strip** – An object that implements the Neopixel or Dotstar protocol.
- **pixel_ranges** (*iterable*) – Pixel ranges (or individual pixels).
- **individual_pixels** (*bool*) – Whether pixel_ranges are individual pixels.

To use with ranges of pixels:

```
import board
import neopixel
from adafruit_led_animation.helper import PixelMap
pixels = neopixel.NeoPixel(board.D6, 32, auto_write=False)

pixel_wing_horizontal = PixelMap(pixels, [(0, 8), (8, 16), (16, 24), (24, 32)])

pixel_wing_horizontal[0] = (255, 255, 0)
pixel_wing_horizontal.show()
```

To use with individual pixels:

```
import board
import neopixel
from adafruit_led_animation.helper import PixelMap
pixels = neopixel.NeoPixel(board.D6, 32, auto_write=False)

pixel_wing_vertical = PixelMap(pixels, [
    (0, 8, 16, 24),
    (1, 9, 17, 25),
    (2, 10, 18, 26),
    (3, 11, 19, 27),
    (4, 12, 20, 28),
    (5, 13, 21, 29),
    (6, 14, 22, 30),
    (7, 15, 23, 31),
], individual_pixels=True)

pixel_wing_vertical[0] = (255, 255, 0)
pixel_wing_vertical.show()
```

auto_write

auto_write from the underlying strip.

brightness

brightness from the underlying strip.

fill (*color*)

Fill the used pixel ranges with color.

Parameters **color** – Color to fill all pixels referenced by this PixelMap definition with.

classmethod horizontal_lines (*pixels, width, height, gridmapper*)

Generate a PixelMap of horizontal lines on a strip arranged in a grid.

Parameters

- **pixels** – pixel object
- **width** – width of grid
- **height** – height of grid
- **gridmapper** – a function to map x and y coordinates to the grid see `vertical_strip_gridmap` and `horizontal_strip_gridmap`

Returns PixelMap

Example: Horizontal lines on a 16x16 grid with the pixel rows oriented vertically, alternating direction every row.

```
PixelMap.horizontal_lines(pixels, 16, 16, vertical_strip_gridmap(16))
```

show ()

Shows the pixels on the underlying strip.

classmethod vertical_lines (*pixels, width, height, gridmapper*)

Generate a PixelMap of horizontal lines on a strip arranged in a grid.

Parameters

- **pixels** – pixel object

- **width** – width of grid
- **height** – height of grid
- **gridmapper** – a function to map x and y coordinates to the grid see `vertical_strip_gridmap` and `horizontal_strip_gridmap`

Returns PixelMap

Example: Vertical lines on a 32x8 grid with the pixel rows oriented vertically, alternating direction every row.

```
PixelMap.vertical_lines(pixels, 32, 8, vertical_strip_gridmap(8))
```

class `adafruit_led_animation.helper.PixelSubset` (*strip, start, end*)
PixelSubset lets you work with a subset of a pixel object.

Parameters

- **strip** – An object that implements the Neopixel or Dotstar protocol.
- **start** (*int*) – Starting pixel number.
- **end** (*int*) – Ending pixel number.

```
import board
import neopixel
from adafruit_led_animation.helper import PixelSubset
pixels = neopixel.NeoPixel(board.D12, 307, auto_write=False)

star_start = 260
star_arm = PixelSubset(pixels, star_start + 7, star_start + 15)
star_arm.fill((255, 0, 255))
pixels.show()
```

auto_write

auto_write from the underlying strip.

brightness

brightness from the underlying strip.

fill (*color*)

Fill the used pixel ranges with color.

show ()

Shows the pixels on the underlying strip.

`adafruit_led_animation.helper.horizontal_strip_gridmap` (*width, alternating=True*)
Determines the pixel number for a grid with strips arranged horizontally.

Parameters

- **width** – strip width in pixels
- **alternating** – strips alternate directions in a zigzag

Returns mapper(x, y)

`adafruit_led_animation.helper.pulse_generator` (*period: float, animation_object, white=False*)

Generates a sequence of colors for a pulse, based on the time period specified. :param period: Pulse duration in seconds. :param animation_object: An animation object to interact with. :param white: Whether the pixel strip has a white pixel.

`adafruit_led_animation.helper.vertical_strip_gridmap` (*height*, *alternating=True*)

Returns a function that determines the pixel number for a grid with strips arranged vertically.

Parameters

- **height** – strip height in pixels
- **alternating** – strips alternate directions in a zigzag

Returns `mapper(x, y)`

6.4 `adafruit_led_animation.group`

Animation group helper for CircuitPython helper library for LED animations..

- Author(s): Roy Hooper, Kattni Rembor

6.4.1 Implementation Notes

Hardware:

- [Adafruit NeoPixels](#)
- [Adafruit DotStars](#)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

class `adafruit_led_animation.group.AnimationGroup` (**members*, *sync=False*,
name=None)

A group of animations that are active together. An example would be grouping a strip of pixels connected to a board and the onboard LED.

Parameters

- **members** – The animation objects or groups.
- **sync** (*bool*) – Synchronises the timing of all members of the group to the settings of the first member of the group. Defaults to `False`.

add_cycle_complete_receiver (*callback*)

Adds an additional callback when the cycle completes.

Parameters **callback** – Additional callback to trigger when a cycle completes. The callback is passed the animation object instance.

animate ()

Call `animate()` from your code's main loop. It will draw all of the animations in the group.

Returns `True` if any animation draw cycle was triggered, otherwise `False`.

color

Use this property to change the color of all members of the animation group.

cycle_complete ()

Called by some animations when they complete an animation cycle. Animations that support cycle complete notifications will have `X` property set to `False`. Override as needed.

cycle_count = `None`

Number of animation cycles completed.

draw_count = None
Number of animation frames drawn.

fill (color)
Fills all pixel objects in the group with a color.

freeze ()
Freeze all animations in the group.

notify_cycles = None
Number of cycles to trigger additional cycle_done notifications after

reset ()
Resets the animations in the group.

resume ()
Resume all animations in the group.

show ()
Draws the current animation group members.

6.5 adafruit_led_animation.sequence

Animation sequence helper for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.5.1 Implementation Notes

Hardware:

- [Adafruit NeoPixels](#)
- [Adafruit DotStars](#)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.sequence.AnimationSequence (*members,          ad-  
                                                         vance_interval=None,  
                                                         auto_clear=True,      ran-  
                                                         dom_order=False,  
                                                         auto_reset=False,    ad-  
                                                         vance_on_cycle_complete=False,  
                                                         name=None)
```

A sequence of Animations to run in succession, looping forever. Advances manually, or at the specified interval.

Parameters

- **members** – The animation objects or groups.
- **advance_interval (int)** – Time in seconds between animations if cycling automatically. Defaults to None.
- **auto_clear (bool)** – Clear the pixels between animations. If True, the current animation will be cleared from the pixels before the next one starts. Defaults to False.
- **random_order (bool)** – Activate the animations in a random order. Defaults to False.

- **auto_reset** (*bool*) – Automatically call reset() on animations when changing animations.
- **advance_on_cycle_complete** (*bool*) – Automatically advance when *cycle_complete* is triggered on member animations. All Animations must support cycle_complete to use this.

```
import board
import neopixel
from adafruit_led_animation.sequence import AnimationSequence
import adafruit_led_animation.animation.comet as comet_animation
import adafruit_led_animation.animation.sparkle as sparkle_animation
import adafruit_led_animation.animation.blink as blink_animation
import adafruit_led_animation.color as color

strip_pixels = neopixel.NeoPixel(board.A1, 30, brightness=1, auto_write=False)

blink = blink_animation.Blink(strip_pixels, 0.2, color.RED)
comet = comet_animation.Comet(strip_pixels, 0.1, color.BLUE)
sparkle = sparkle_animation.Sparkle(strip_pixels, 0.05, color.GREEN)

animations = AnimationSequence(blink, comet, sparkle, advance_interval=5)

while True:
    animations.animate()
```

activate (*index*)

Activates a specific animation.

add_cycle_complete_receiver (*callback*)

Adds an additional callback when the cycle completes.

Parameters *callback* – Additional callback to trigger when a cycle completes. The callback is passed the animation object instance.

animate ()

Call animate() from your code's main loop. It will draw the current animation or go to the next animation based on the advance_interval if set.

Returns True if the animation draw cycle was triggered, otherwise False.

color

Use this property to change the color of all animations in the sequence.

current_animation

Returns the current animation in the sequence.

cycle_complete ()

Called by some animations when they complete an animation cycle. Animations that support cycle complete notifications will have X property set to False. Override as needed.

fill (*color*)

Fills the current animation with a color.

freeze ()

Freeze the current animation in the sequence. Also stops auto_advance.

next ()

Jump to the next animation.

random ()

Jump to a random animation.

reset()

Resets the current animation.

resume()

Resume the current animation in the sequence, and resumes auto advance if enabled.

show()

Draws the current animation group members.

6.6 adafruit_led_animation.animation.blink

Blink animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.6.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

class adafruit_led_animation.animation.blink.**Blink**(*pixel_object*, *speed*, *color*,
name=None)

Blink a color on and off.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation speed in seconds, e.g. 0.1.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.

6.7 adafruit_led_animation.animation.solid

Solid animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.7.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

class adafruit_led_animation.animation.solid.**Solid** (*pixel_object*, *color*, *name=None*)
A solid color.

Parameters

- **pixel_object** – The initialised LED object.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.

6.8 adafruit_led_animation.animation.colorcycle

Color cycle animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.8.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

class adafruit_led_animation.animation.colorcycle.**ColorCycle** (*pixel_object*, *speed*,
colors=((255, 0,
0), (255, 40, 0),
(255, 150, 0), (0,
255, 0), (0, 0, 255),
(180, 0, 255)),
name=None)

Animate a sequence of one or more colors, cycling at the specified speed.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation speed in seconds, e.g. 0.1.
- **colors** – A list of colors to cycle through in (r, g, b) tuple, or 0x000000 hex format. Defaults to a rainbow color cycle.

draw()

Animation subclasses must implement draw() to render the animation sequence. Draw must call show().

reset()

Resets to the first color.

6.9 adafruit_led_animation.animation.chase

Theatre chase animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.9.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.chase.Chase (pixel_object, speed, color, size=2,  
                                                    spacing=3,          reverse=False,  
                                                    name=None)
```

Chase pixels in one direction in a single color, like a theater marquee sign.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation speed rate in seconds, e.g. 0.1.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.
- **size** – Number of pixels to turn on in a row.
- **spacing** – Number of pixels to turn off in a row.
- **reverse** – Reverse direction of movement.

```
bar_color (n, pixel_no=0)
```

Generate the color for the n'th bar_color in the Chase

Parameters

- **n** – The pixel group to get the color for
- **pixel_no** – Which pixel in the group to get the color for

```
draw ()
```

Animation subclasses must implement draw() to render the animation sequence. Draw must call show().

```
reset ()
```

Reset the animation.

```
reverse
```

Whether the animation is reversed

```
space_color (n, pixel_no=0)
```

Generate the spacing color for the n'th bar_color in the Chase

Parameters

- **n** – The pixel group to get the spacing color for
- **pixel_no** – Which pixel in the group to get the spacing color for

6.10 adafruit_led_animation.animation.comet

Comet animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.10.1 Implementation Notes

Hardware:

- [Adafruit NeoPixels](#)
- [Adafruit DotStars](#)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.comet.Comet (pixel_object,    speed,    color,  
                                                    tail_length=0,    reverse=False,  
                                                    bounce=False, name=None)
```

A comet animation.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation speed in seconds, e.g. 0.1.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.
- **tail_length** (*int*) – The length of the comet. Defaults to 25% of the length of the pixel_object. Automatically compensates for a minimum of 2 and a maximum of the length of the pixel_object.
- **reverse** (*bool*) – Animates the comet in the reverse order. Defaults to False.
- **bounce** (*bool*) – Comet will bounce back and forth. Defaults to True.

draw()

Animation subclasses must implement draw() to render the animation sequence. Draw must call show().

reset()

Resets to the first color.

6.11 adafruit_led_animation.animation.pulse

Pulse animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.11.1 Implementation Notes

Hardware:

- [Adafruit NeoPixels](#)
- [Adafruit DotStars](#)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.pulse.Pulse (pixel_object, speed, color, pe-  
                                                    riod=5, name=None)
```

Pulse all pixels a single color.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation refresh rate in seconds, e.g. 0.1.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.
- **period** – Period to pulse the LEDs over. Default 5.

draw()

Animation subclasses must implement draw() to render the animation sequence. Draw must call show().

reset()

Resets the animation.

6.12 adafruit_led_animation.animation.rainbow

Rainbow animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.12.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.rainbow.Rainbow(pixel_object,      speed,  
                                                         period=5,      step=1,  
                                                         name=None,      precom-  
                                                         pute_rainbow=True)
```

The classic rainbow color wheel.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation refresh rate in seconds, e.g. 0.1.
- **period** (*float*) – Period to cycle the rainbow over in seconds. Default 5.
- **step** (*float*) – Color wheel step. Default 1.
- **name** (*str*) – Name of animation (optional, useful for sequences and debugging).
- **precompute_rainbow** (*bool*) – Whether to precompute the rainbow. Uses more memory. (default True).

draw()

Animation subclasses must implement draw() to render the animation sequence. Draw must call show().

generate_rainbow()

Generates the rainbow.

reset()

Resets the animation.

6.13 adafruit_led_animation.animation.sparkle

Sparkle animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.13.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.sparkle.Sparkle(pixel_object, speed,  
                                                    color, num_sparkles=1,  
                                                    name=None)
```

Sparkle animation of a single color.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation speed in seconds, e.g. 0.1.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.
- **num_sparkles** – Number of sparkles to generate per animation cycle.

draw()

Animation subclasses must implement draw() to render the animation sequence. Draw must call show().

6.14 adafruit_led_animation.animation.rainbowchase

Rainbow chase animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.14.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.rainbowchase.RainbowChase (pixel_object,  
                                                                speed,  
                                                                size=2, spacing=3, reverse=False,  
                                                                name=None,  
                                                                wheel_step=8)
```

Chase pixels in one direction, like a theater marquee but with rainbows!

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation speed rate in seconds, e.g. 0.1.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.
- **size** – Number of pixels to turn on in a row.
- **spacing** – Number of pixels to turn off in a row.
- **reverse** – Reverse direction of movement.
- **wheel_step** – How many colors to skip in *colorwheel* per bar (default 8)

bar_color (*n*, *pixel_no*=0)

Generate the color for the n'th bar_color in the Chase

Parameters

- **n** – The pixel group to get the color for
- **pixel_no** – Which pixel in the group to get the color for

cycle_complete ()

Called by some animations when they complete an animation cycle. Animations that support cycle complete notifications will have X property set to False. Override as needed.

6.15 adafruit_led_animation.animation.rainbowcomet

Rainbow comet for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.15.1 Implementation Notes

Hardware:

- Adafruit NeoPixels
- Adafruit DotStars

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.rainbowcomet.RainbowComet (pixel_object,  
                                                                speed,  
                                                                tail_length=10,  
                                                                re-  
                                                                verse=False,  
                                                                bounce=False,  
                                                                color-  
                                                                wheel_offset=0,  
                                                                name=None)
```

A rainbow comet animation.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation speed in seconds, e.g. 0.1.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.
- **tail_length** (*int*) – The length of the comet. Defaults to 10. Cannot exceed the number of pixels present in the pixel object, e.g. if the strip is 30 pixels long, the `tail_length` cannot exceed 30 pixels.
- **reverse** (*bool*) – Animates the comet in the reverse order. Defaults to False.
- **bounce** (*bool*) – Comet will bounce back and forth. Defaults to True.
- **colorwheel_offset** (*int*) – Offset from start of colorwheel (0-255).

6.16 adafruit_led_animation.animation.rainbowsparkle

Rainbow sparkle for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, Kattni Rembor

6.16.1 Implementation Notes

Hardware:

- [Adafruit NeoPixels](#)
- [Adafruit DotStars](#)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.rainbowsparkle.RainbowSparkle (pixel_object,  
                                                                speed,  
                                                                pe-  
                                                                riod=5,  
                                                                num_sparkles=None,  
                                                                step=1,  
                                                                name=None,  
                                                                back-  
                                                                ground_brightness=0.2)
```

Rainbow sparkle animation.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*float*) – Animation refresh rate in seconds, e.g. 0.1.
- **period** (*float*) – Period to cycle the rainbow over in seconds. Default 5.
- **num_sparkles** (*int*) – The number of sparkles to display. Defaults to 1/20 of the pixel object length.
- **step** (*float*) – Color wheel step. Default 1.
- **name** (*str*) – Name of animation (optional, useful for sequences and debugging).
- **background_brightness** (*float*) – The brightness of the background rainbow. Defaults to 0.2 or 20 percent.
- **precompute_rainbow** (*bool*) – Whether to precompute the rainbow. Uses more memory. (default True).

generate_rainbow()
Generates the rainbow.

show()
Displays the updated pixels. Called during animates with changes.

6.17 adafruit_led_animation.animation.sparklepulse

Sparkle-pulse animation for CircuitPython helper library for LED animations.

- Author(s): Roy Hooper, dmolavi

6.17.1 Implementation Notes

Hardware:

- [Adafruit NeoPixels](#)
- [Adafruit DotStars](#)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

```
class adafruit_led_animation.animation.sparklepulse.SparklePulse (pixel_object,  
                                                                speed, color,  
                                                                period=5,  
                                                                max_intensity=1,  
                                                                min_intensity=0)
```

Combination of the Spark and Pulse animations.

Parameters

- **pixel_object** – The initialised LED object.
- **speed** (*int*) – Animation refresh rate in seconds, e.g. 0.1.
- **color** – Animation color in (r, g, b) tuple, or 0x000000 hex format.
- **period** – Period to pulse the LEDs over. Default 5.
- **max_intensity** – The maximum intensity to pulse, between 0 and 1.0. Default 1.
- **min_intensity** – The minimum intensity to pulse, between 0 and 1.0. Default 0.

draw ()

Animation subclasses must implement draw() to render the animation sequence. Draw must call show().

CHAPTER 7

Indices and tables

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

a

- `adafruit_led_animation.animation`, [13](#)
- `adafruit_led_animation.animation.blink`,
[21](#)
- `adafruit_led_animation.animation.chase`,
[22](#)
- `adafruit_led_animation.animation.colorcycle`,
[22](#)
- `adafruit_led_animation.animation.comet`,
[23](#)
- `adafruit_led_animation.animation.pulse`,
[24](#)
- `adafruit_led_animation.animation.rainbow`,
[25](#)
- `adafruit_led_animation.animation.rainbowchase`,
[26](#)
- `adafruit_led_animation.animation.rainbowcomet`,
[27](#)
- `adafruit_led_animation.animation.rainbowsparkle`,
[28](#)
- `adafruit_led_animation.animation.solid`,
[21](#)
- `adafruit_led_animation.animation.sparkle`,
[25](#)
- `adafruit_led_animation.animation.sparklepulse`,
[29](#)
- `adafruit_led_animation.color`, [15](#)
- `adafruit_led_animation.group`, [18](#)
- `adafruit_led_animation.helper`, [15](#)
- `adafruit_led_animation.sequence`, [19](#)

A

`activate()` (*adafruit_led_animation.sequence.AnimationSequence* *method*), 20
`adafruit_led_animation.animation` (*module*), 13
`adafruit_led_animation.animation.blink` (*module*), 21
`adafruit_led_animation.animation.chase` (*module*), 22
`adafruit_led_animation.animation.colorcycle` (*module*), 22
`adafruit_led_animation.animation.comet` (*module*), 23
`adafruit_led_animation.animation.pulse` (*module*), 24
`adafruit_led_animation.animation.rainbow` (*module*), 25
`adafruit_led_animation.animation.rainbowchase` (*module*), 26
`adafruit_led_animation.animation.rainbowcomet` (*module*), 27
`adafruit_led_animation.animation.rainbowspace` (*module*), 28
`adafruit_led_animation.animation.solid` (*module*), 21
`adafruit_led_animation.animation.sparkle` (*module*), 25
`adafruit_led_animation.animation.sparklepulse` (*module*), 29
`adafruit_led_animation.color` (*module*), 15
`adafruit_led_animation.group` (*module*), 18
`adafruit_led_animation.helper` (*module*), 15
`adafruit_led_animation.sequence` (*module*), 19
`add_cycle_complete_receiver()` (*adafruit_led_animation.animation.AnimationSequence* *method*), 14
`add_cycle_complete_receiver()` (*adafruit_led_animation.group.AnimationGroup* *method*), 18
`add_cycle_complete_receiver()` (*adafruit_led_animation.sequence.AnimationSequence* *method*), 20
`animate()` (*adafruit_led_animation.animation.Animation* *method*), 14
`animate()` (*adafruit_led_animation.group.AnimationGroup* *method*), 18
`animate()` (*adafruit_led_animation.sequence.AnimationSequence* *method*), 20
`Animation` (*class* in *adafruit_led_animation.animation*), 14
`AnimationGroup` (*class* in *adafruit_led_animation.group*), 18
`AnimationSequence` (*class* in *adafruit_led_animation.sequence*), 19
`auto_write` (*adafruit_led_animation.helper.PixelMap* *attribute*), 16
`auto_write` (*adafruit_led_animation.helper.PixelSubset* *attribute*), 17

B

`bar_color()` (*adafruit_led_animation.animation.chase.Chase* *method*), 23
`bar_color()` (*adafruit_led_animation.animation.rainbowchase.Rainbowchase* *method*), 27
`Blink` (*class* in *adafruit_led_animation.animation.blink*), 21
`brightness` (*adafruit_led_animation.helper.PixelMap* *attribute*), 16
`brightness` (*adafruit_led_animation.helper.PixelSubset* *attribute*), 17

C

`Chase` (*class* in *adafruit_led_animation.animation.chase*), 23
`color` (*adafruit_led_animation.animation.Animation* *attribute*), 14
`color` (*adafruit_led_animation.group.AnimationGroup* *attribute*), 18

`color (adafruit_led_animation.sequence.AnimationSequence fill () (adafruit_led_animation.helper.PixelSubset attribute), 20 method), 17`

`ColorCycle (class in fill () (adafruit_led_animation.sequence.AnimationSequence adafruit_led_animation.animation.colorcycle), method), 20`

`22 freeze () (adafruit_led_animation.animation.Animation method), 14`

`colorwheel () (in module method), 14`

`Comet (class in adafruit_led_animation.animation.comet), method), 19`

`24 freeze () (adafruit_led_animation.sequence.AnimationSequence method), 20`

`current_animation (adafruit_led_animation.sequence.AnimationSequence attribute), 20`

G

`cycle_complete () (adafruit_led_animation.animation.Animation generate_rainbow () (adafruit_led_animation.animation.rainbow.Rainbow method), 14 method), 25`

`cycle_complete () (adafruit_led_animation.animation.rainbowchase.RainbowChase generate_rainbow () (adafruit_led_animation.animation.rainbowsparkle.RainbowSparkle method), 27 method), 29`

`cycle_complete () (adafruit_led_animation.group.AnimationGroup generate_rainbow () (adafruit_led_animation.animation.rainbowsparkle.RainbowSparkle method), 18 method), 29`

`cycle_complete () (adafruit_led_animation.sequence.AnimationSequence generate_rainbow () (adafruit_led_animation.animation.rainbowsparkle.RainbowSparkle method), 20 method), 20`

H

`cycle_count (adafruit_led_animation.animation.Animation horizontal_lines () (adafruit_led_animation.helper.PixelMap attribute), 14 class method), 16`

`cycle_count (adafruit_led_animation.group.AnimationGroup horizontal_lines () (adafruit_led_animation.helper.PixelMap attribute), 18 class method), 16`

`horizontal_strip_gridmap () (in module adafruit_led_animation.helper), 17`

D

`draw () (adafruit_led_animation.animation.Animation method), 14`

`draw () (adafruit_led_animation.animation.chase.Chase method), 23`

`draw () (adafruit_led_animation.animation.colorcycle.ColorCycle attribute), 14`

`method), 22`

`draw () (adafruit_led_animation.animation.comet.Comet attribute), 19`

`method), 24`

`draw () (adafruit_led_animation.animation.pulse.Pulse attribute), 14`

`method), 25`

`draw () (adafruit_led_animation.animation.rainbow.Rainbow attribute), 14`

`method), 25`

`draw () (adafruit_led_animation.animation.sparkle.Sparkle attribute), 14`

`method), 26`

`draw () (adafruit_led_animation.animation.sparklepulse.SparklePulse attribute), 14`

`method), 30`

`draw_count (adafruit_led_animation.animation.Animation attribute), 14`

`draw_count (adafruit_led_animation.group.AnimationGroup attribute), 18`

N

`next () (adafruit_led_animation.sequence.AnimationSequence method), 20`

`notify_cycles (adafruit_led_animation.animation.Animation attribute), 14`

`notify_cycles (adafruit_led_animation.group.AnimationGroup attribute), 19`

P

`peers (adafruit_led_animation.animation.Animation attribute), 14`

`PixelMap (class in adafruit_led_animation.helper), 15`

`PixelSubset (class in adafruit_led_animation.helper), 17`

`SparklePulse (class in adafruit_led_animation.animation.pulse), 24`

`pulse_generator () (in module adafruit_led_animation.helper), 17`

R

`Rainbow (class in adafruit_led_animation.animation.rainbow), 25`

`RainbowChase (class in adafruit_led_animation.animation.rainbowchase), 26`

`RainbowComet (class in adafruit_led_animation.animation.rainbowcomet), 27`

F

`fill () (adafruit_led_animation.animation.Animation method), 14`

`fill () (adafruit_led_animation.group.AnimationGroup method), 19`

`fill () (adafruit_led_animation.helper.PixelMap method), 16`

```

RainbowSparkle      (class      in V
    adafruit_led_animation.animation.rainbowsparkle)
28      vertical_lines() (adafruit_led_animation.helper.PixelMap
    class method), 16
random() (adafruit_led_animation.sequence.AnimationSequence
    method), 20      vertical_strip_gridmap() (in module
    adafruit_led_animation.helper), 17
reset() (adafruit_led_animation.animation.Animation
    method), 14
reset() (adafruit_led_animation.animation.chase.Chase
    method), 23
reset() (adafruit_led_animation.animation.colorcycle.ColorCycle
    method), 22
reset() (adafruit_led_animation.animation.comet.Comet
    method), 24
reset() (adafruit_led_animation.animation.pulse.Pulse
    method), 25
reset() (adafruit_led_animation.animation.rainbow.Rainbow
    method), 25
reset() (adafruit_led_animation.group.AnimationGroup
    method), 19
reset() (adafruit_led_animation.sequence.AnimationSequence
    method), 21
resume() (adafruit_led_animation.animation.Animation
    method), 15
resume() (adafruit_led_animation.group.AnimationGroup
    method), 19
resume() (adafruit_led_animation.sequence.AnimationSequence
    method), 21
reverse(adafruit_led_animation.animation.chase.Chase
    attribute), 23

```

S

```

show() (adafruit_led_animation.animation.Animation
    method), 15
show() (adafruit_led_animation.animation.rainbowsparkle.RainbowSparkle
    method), 29
show() (adafruit_led_animation.group.AnimationGroup
    method), 19
show() (adafruit_led_animation.helper.PixelMap
    method), 16
show() (adafruit_led_animation.helper.PixelSubset
    method), 17
show() (adafruit_led_animation.sequence.AnimationSequence
    method), 21
Solid(class in adafruit_led_animation.animation.solid),
21
space_color() (adafruit_led_animation.animation.chase.Chase
    method), 23
Sparkle(class in adafruit_led_animation.animation.sparkle),
26
SparklePulse      (class      in
    adafruit_led_animation.animation.sparklepulse),
29
speed (adafruit_led_animation.animation.Animation
    attribute), 15

```