

---

# **LED***AnimationLibraryDocumentation*

***Release 1.0***

**Kattni Rembor**

**Mar 03, 2021**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Installing from PyPI</b>	<b>5</b>
<b>3</b>	<b>Usage Example</b>	<b>7</b>
<b>4</b>	<b>Contributing</b>	<b>9</b>
<b>5</b>	<b>Building locally</b>	<b>11</b>
5.1	Zip release files . . . . .	11
5.2	Sphinx documentation . . . . .	11
<b>6</b>	<b>Table of Contents</b>	<b>13</b>
6.1	Simple test . . . . .	13
6.2	adafruit_led_animation.animation . . . . .	14
6.2.1	Implementation Notes . . . . .	14
6.3	adafruit_led_animation.color . . . . .	15
6.3.1	Implementation Notes . . . . .	15
6.4	adafruit_led_animation.helper . . . . .	16
6.4.1	Implementation Notes . . . . .	17
6.5	adafruit_led_animation.group . . . . .	20
6.5.1	Implementation Notes . . . . .	20
6.6	adafruit_led_animation.sequence . . . . .	21
6.6.1	Implementation Notes . . . . .	21
6.7	adafruit_led_animation.animation.blink . . . . .	23
6.7.1	Implementation Notes . . . . .	23
6.8	adafruit_led_animation.animation.solid . . . . .	24
6.8.1	Implementation Notes . . . . .	24
6.9	adafruit_led_animation.animation.colorcycle . . . . .	24
6.9.1	Implementation Notes . . . . .	24
6.10	adafruit_led_animation.animation.chase . . . . .	25
6.10.1	Implementation Notes . . . . .	25
6.11	adafruit_led_animation.animation.comet . . . . .	26
6.11.1	Implementation Notes . . . . .	26
6.12	adafruit_led_animation.animation.pulse . . . . .	27
6.12.1	Implementation Notes . . . . .	27
6.13	adafruit_led_animation.animation.rainbow . . . . .	28

6.13.1	Implementation Notes	28
6.14	<code>adafruit_led_animation.animation.sparkle</code>	28
6.14.1	Implementation Notes	29
6.15	<code>adafruit_led_animation.animation.rainbowchase</code>	29
6.15.1	Implementation Notes	29
6.16	<code>adafruit_led_animation.animation.rainbowcomet</code>	30
6.16.1	Implementation Notes	30
6.17	<code>adafruit_led_animation.animation.rainbowsparkle</code>	31
6.17.1	Implementation Notes	31
6.18	<code>adafruit_led_animation.animation.sparklepulse</code>	32
6.18.1	Implementation Notes	32
<b>7</b>	<b>Indices and tables</b>	<b>35</b>
	<b>Python Module Index</b>	<b>37</b>
	<b>Index</b>	<b>39</b>

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  # SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """
5  This simpletest example displays the Blink animation.
6
7  For NeoPixel FeatherWing. Update pixel_pin and pixel_num to match your wiring if using
8  a different form of NeoPixels.
9  """
10 import board
11 import neopixel
12 from adafruit_led_animation.animation.blink import Blink
13 from adafruit_led_animation.color import RED
14
15 # Update to match the pin connected to your NeoPixels
16 pixel_pin = board.D6
17 # Update to match the number of NeoPixels you have connected
18 pixel_num = 32
19
20 pixels = neopixel.NeoPixel(pixel_pin, pixel_num, brightness=0.5, auto_write=False)
21
22 blink = Blink(pixels, speed=0.5, color=RED)
23
24 while True:
25     blink.animate()
```

## 6.2 adafruit\_led\_animation.animation

Animation base class for CircuitPython helper library for LED animations.

- Author(s): 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.

**after\_draw** ()

Animation subclasses may implement after\_draw() to do operations after the main draw() is called.

**animate** (show=True)

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

**Parameters** **show** (bool) – Whether to automatically call show on the pixel object when an animation fires. Default True.

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

**color**

The current color.

**cycle\_count** = None

Number of animation cycles completed.

**draw** ()

Animation subclasses must implement draw() to render the animation sequence. Animations should not call show(), as animate() will do so, after after\_draw(). Animations should set .cycle\_done = True when an animation cycle is completed.

**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

**on\_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.

**peers**

Get the animation's peers. Peers are drawn, then shown together.

**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.

## 6.3 adafruit\_led\_animation.color

Color variables assigned to RGB values made available for import.

- Author(s): 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>

```
adafruit_led_animation.color.AMBER = (255, 100, 0)
    Amber.
```

```
adafruit_led_animation.color.AQUA = (50, 255, 255)
    Aqua.
```

```
adafruit_led_animation.color.BLACK = (0, 0, 0)
    Black or off.
```

```
adafruit_led_animation.color.BLUE = (0, 0, 255)
    Blue.
```

```
adafruit_led_animation.color.CYAN = (0, 255, 255)
    Cyan.
```

```
adafruit_led_animation.color.GOLD = (255, 222, 30)
    Gold.
```

```
adafruit_led_animation.color.GREEN = (0, 255, 0)
    Green.
```

```
adafruit_led_animation.color.JADE = (0, 255, 40)
    Jade.
adafruit_led_animation.color.MAGENTA = (255, 0, 20)
    Magenta.
adafruit_led_animation.color.OLD_LACE = (253, 245, 230)
    Old lace or warm white.
adafruit_led_animation.color.ORANGE = (255, 40, 0)
    Orange.
adafruit_led_animation.color.PINK = (242, 90, 255)
    Pink.
adafruit_led_animation.color.PURPLE = (180, 0, 255)
    Purple.
adafruit_led_animation.color.RAINBOW = ((255, 0, 0), (255, 40, 0), (255, 150, 0), (0, 255,
    RAINBOW is a list of colors to use for cycling through. Includes, in order: red, orange, yellow, green, blue, and
    purple.
adafruit_led_animation.color.RED = (255, 0, 0)
    Red.
adafruit_led_animation.color.RGBW_WHITE_RGB = (255, 255, 255, 0)
    RGBW_WHITE_RGB is for RGBW strips to illuminate only the RGB diodes.
adafruit_led_animation.color.RGBW_WHITE_RGBW = (255, 255, 255, 255)
    RGBW_WHITE_RGBW is for RGBW strips to illuminate the RGB and White diodes.
adafruit_led_animation.color.RGBW_WHITE_W = (0, 0, 0, 255)
    RGBW_WHITE_W is for RGBW strips to illuminate only White diode.
adafruit_led_animation.color.TEAL = (0, 255, 120)
    Teal.
adafruit_led_animation.color.WHITE = (255, 255, 255)
    White.
adafruit_led_animation.color.YELLOW = (255, 150, 0)
    Yellow.
adafruit_led_animation.color.calculate_intensity(color, intensity=1.0)
    Takes a RGB[W] color tuple and adjusts the intensity. :param float intensity: :param color: color value (tuple,
    list or int) :return: color
adafruit_led_animation.color.colorwheel(pos)
    Colorwheel is built into CircuitPython's _pixelbuf. A separate colorwheel is included here for use with Circuit-
    Python builds that do not include _pixelbuf, as with some of the SAMD21 builds. To use: input a value 0 to 255
    to get a color value. The colours are a transition from red to green to blue and back to red.
```

## 6.4 `adafruit_led_animation.helper`

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

- Author(s): 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.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 groups of 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()
```

To use with individual pixels:

```
import board
import neopixel
```

(continues on next page)

(continued from previous page)

```

import time
from adafruit_led_animation.helper import PixelMap

pixels = neopixel.NeoPixel(board.D6, 8, auto_write=False)

pixel_map = PixelMap(pixels, [
    0, 7, 1, 6, 2, 5, 3, 4
], individual_pixels=True)

n = 0
while True:
    pixel_map[n] = AMBER
    pixel_map.show()
    n = n + 1
    if n > 7:
        n = 0
        pixel_map.fill(0)
    time.sleep(0.25)

```

**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** (*pixel\_object, width, height, gridmap*)

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

**Parameters**

- **pixel\_object** – pixel object
- **width** – width of grid
- **height** – height of grid
- **gridmap** – 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** (*pixel\_object, width, height, gridmap*)

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

**Parameters**

- **pixel\_object** – pixel object

- **width** – width of grid
- **height** – height of grid
- **gridmap** – 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` (*pixel\_object*, *start*, *end*)  
PixelSubset lets you work with a subset of a pixel object.

#### Parameters

- **pixel\_object** – 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()
```

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

#### Parameters

- **width** – grid width in pixels
- **alternating** – Whether or not the lines in the grid run alternate directions in a zigzag

**Returns** `mapper(x, y)`

`adafruit_led_animation.helper.pulse_generator` (*period: float*, *animation\_object*, *dotstar\_pwm=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 *dotstar\_pwm*: Whether to use the dotstar per pixel PWM value for brightness control.

`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** – grid height in pixels
- **alternating** – Whether or not the lines in the grid run alternate directions in a zigzag

**Returns** `mapper(x, y)`

## 6.5 adafruit\_led\_animation.group

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

- Author(s): 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.group.AnimationGroup (*members, sync=False,  
                                                    name=None)
```

AnimationGroup synchronizes multiple animations. Allows for multiple animations to be kept in sync, whether or not the same animation or pixel object is in use.

#### Parameters

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

#### Example

```
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 (show=True)
```

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_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



**on\_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.

**reset()**

Resets the animations in the group.

**resume()**

Resume all animations in the group.

**show()**

Draws the current animation group members.

## 6.6 adafruit\_led\_animation.sequence

Animation sequence helper for CircuitPython helper library for LED animations.

- Author(s): 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.sequence.AnimateOnce(*members, **kwargs)`

Wrapper around `AnimationSequence` that returns `False` to `animate()` until a sequence has completed. Takes the same arguments as `AnimationSequence`, but overrides `advance_on_cycle_complete=True` and `advance_interval=0`

Example:

This example animates a comet in one direction then pulses red momentarily

```
import board
import neopixel
from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.animation.pulse import Pulse
from adafruit_led_animation.color import BLUE, RED
from adafruit_led_animation.sequence import AnimateOnce

strip_pixels = neopixel.NeoPixel(board.A1, 30, brightness=0.5, auto_write=False)

comet = Comet(strip_pixels, 0.01, color=BLUE, bounce=False)
pulse = Pulse(strip_pixels, 0.01, color=RED, period=2)

animations = AnimateOnce(comet, pulse)

while animations.animate():
    pass
```

**animate** (*show=True*)

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.

**on\_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.

```
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 `on_cycle_complete` is triggered on member animations. All Animations must support `on_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** (*show=True*)

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.

**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.

**on\_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.

**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.

## 6.7 adafruit\_led\_animation.animation.blink

Blink animation for CircuitPython helper library for LED animations.

- Author(s): 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.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.

## 6.8 adafruit\_led\_animation.animation.solid

Solid animation for CircuitPython helper library for LED animations.

- Author(s): 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.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.

## 6.9 adafruit\_led\_animation.animation.colorcycle

Color cycle animation for CircuitPython helper library for LED animations.

- Author(s): 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.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. Animations should not call show(), as animate() will do so, after after\_draw(). Animations should set .cycle\_done = True when an animation cycle is completed.

#### **reset()**

Resets to the first color.

## 6.10 adafruit\_led\_animation.animation.chase

Theatre chase animation for CircuitPython helper library for LED animations.

- Author(s): 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.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. Animations should not call show(), as animate() will do so, after after\_draw(). Animations should set .cycle\_done = True when an animation cycle is completed.

**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

## 6.11 adafruit\_led\_animation.animation.comet

Comet animation for CircuitPython helper library for LED animations.

- Author(s): 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.comet.Comet (pixel_object, speed, color,  
                                                    tail_length=0, reverse=False,  
                                                    bounce=False, name=None,  
                                                    ring=False)
```

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`.
- **ring** (*bool*) – Ring mode. Defaults to `False`.

**draw()**

Animation subclasses must implement `draw()` to render the animation sequence. Animations should not call `show()`, as `animate()` will do so, after `after_draw()`. Animations should set `.cycle_done = True` when an animation cycle is completed.

**reset()**

Resets to the first state.

## 6.12 adafruit\_led\_animation.animation.pulse

Pulse animation for CircuitPython helper library for LED animations.

- Author(s): 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.pulse.Pulse` (*pixel\_object*, *speed*, *color*, *period=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. Animations should not call `show()`, as `animate()` will do so, after `after_draw()`. Animations should set `.cycle_done = True` when an animation cycle is completed.

**reset()**

Resets the animation.

## 6.13 adafruit\_led\_animation.animation.rainbow

Rainbow animation for CircuitPython helper library for LED animations.

- Author(s): 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.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. Animations should not call `show()`, as `animate()` will do so, after `after_draw()`. Animations should set `.cycle_done = True` when an animation cycle is completed.

#### **generate\_rainbow()**

Generates the rainbow.

#### **reset()**

Resets the animation.

## 6.14 adafruit\_led\_animation.animation.sparkle

Sparkle animation for CircuitPython helper library for LED animations.

- Author(s): 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.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.

#### **after\_draw()**

Animation subclasses may implement after\_draw() to do operations after the main draw() is called.

#### **draw()**

Animation subclasses must implement draw() to render the animation sequence. Animations should not call show(), as animate() will do so, after after\_draw(). Animations should set .cycle\_done = True when an animation cycle is completed.

## 6.15 adafruit\_led\_animation.animation.rainbowchase

Rainbow chase animation for CircuitPython helper library for LED animations.

- Author(s): 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.rainbowchase.RainbowChase(pixel_object,  
                                                                speed,  
                                                                size=2, spacing=3, reverse=False,  
                                                                name=None,  
                                                                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.
- **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

**on\_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.

## 6.16 adafruit\_led\_animation.animation.rainbowcomet

Rainbow comet for CircuitPython helper library for LED animations.

- Author(s): 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.rainbowcomet.RainbowComet (pixel_object,
                                                                speed,
                                                                tail_length=10,
                                                                re-
                                                                verse=False,
                                                                bounce=False,
                                                                color-
                                                                wheel_offset=0,
                                                                step=0,
                                                                name=None,
                                                                ring=False)
```

A rainbow comet animation.

**Parameters**

- **pixel\_object** – The initialised LED object.
- **speed** (*float*) – Animation speed in seconds, e.g. 0.1.
- **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).
- **step** (*int*) – Colorwheel step (defaults to automatic).
- **ring** (*bool*) – Ring mode. Defaults to `False`.

## 6.17 adafruit\_led\_animation.animation.rainbowsparkle

Rainbow sparkle for CircuitPython helper library for LED animations.

- Author(s): Kattni Rembor

### 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.rainbowsparkle.RainbowSparkle(pixel_object,  
                                                                    speed,  
                                                                    pe-  
                                                                    riod=5,  
                                                                    num_sparkles=None,  
                                                                    step=1,  
                                                                    name=None,  
                                                                    back-  
                                                                    ground_brightness=0.2,  
                                                                    pre-  
                                                                    com-  
                                                                    pute_rainbow=True)
```

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).

**after\_draw** ()

Animation subclasses may implement after\_draw() to do operations after the main draw() is called.

**generate\_rainbow** ()

Generates the rainbow.

## 6.18 adafruit\_led\_animation.animation.sparklepulse

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

- Author(s): dmlavi

### 6.18.1 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,  
                                                                name=None)
```

Combination of the Sparkle 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.

**after\_draw** ()

Animation subclasses may implement after\_draw() to do operations after the main draw() is called.

**draw()**

Animation subclasses must implement draw() to render the animation sequence. Animations should not call show(), as animate() will do so, after after\_draw(). Animations should set .cycle\_done = True when an animation cycle is completed.



## CHAPTER 7

---

### Indices and tables

---

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





### a

- `adafruit_led_animation.animation`, [13](#)
- `adafruit_led_animation.animation.blink`,  
[23](#)
- `adafruit_led_animation.animation.chase`,  
[25](#)
- `adafruit_led_animation.animation.colorcycle`,  
[24](#)
- `adafruit_led_animation.animation.comet`,  
[26](#)
- `adafruit_led_animation.animation.pulse`,  
[27](#)
- `adafruit_led_animation.animation.rainbow`,  
[27](#)
- `adafruit_led_animation.animation.rainbowchase`,  
[29](#)
- `adafruit_led_animation.animation.rainbowcomet`,  
[30](#)
- `adafruit_led_animation.animation.rainbowsparkle`,  
[31](#)
- `adafruit_led_animation.animation.solid`,  
[24](#)
- `adafruit_led_animation.animation.sparkle`,  
[28](#)
- `adafruit_led_animation.animation.sparklepulse`,  
[32](#)
- `adafruit_led_animation.color`, [15](#)
- `adafruit_led_animation.group`, [19](#)
- `adafruit_led_animation.helper`, [16](#)
- `adafruit_led_animation.sequence`, [21](#)



## A

`activate()` (*adafruit\_led\_animation.sequence.AnimationSequence* *method*), 22  
`adafruit_led_animation.animation` (*module*), 13  
`adafruit_led_animation.animation.blink` (*module*), 23  
`adafruit_led_animation.animation.chase` (*module*), 25  
`adafruit_led_animation.animation.colorcycle` (*module*), 24  
`adafruit_led_animation.animation.comet` (*module*), 26  
`adafruit_led_animation.animation.pulse` (*module*), 27  
`adafruit_led_animation.animation.rainbow` (*module*), 27  
`adafruit_led_animation.animation.rainbowchase` (*module*), 29  
`adafruit_led_animation.animation.rainbowcomet` (*module*), 30  
`adafruit_led_animation.animation.rainbowspace` (*module*), 31  
`adafruit_led_animation.animation.solid` (*module*), 24  
`adafruit_led_animation.animation.sparkle` (*module*), 28  
`adafruit_led_animation.animation.sparklepulse` (*module*), 32  
`adafruit_led_animation.color` (*module*), 15  
`adafruit_led_animation.group` (*module*), 19  
`adafruit_led_animation.helper` (*module*), 16  
`adafruit_led_animation.sequence` (*module*), 21  
`add_cycle_complete_receiver()` (*adafruit\_led\_animation.animation.Animation* *method*), 14  
`add_cycle_complete_receiver()` (*adafruit\_led\_animation.group.AnimationGroup* *method*), 20  
`add_cycle_complete_receiver()` (*adafruit\_led\_animation.sequence.AnimationSequence* *method*), 22  
`after_draw()` (*adafruit\_led\_animation.animation.Animation* *method*), 14  
`after_draw()` (*adafruit\_led\_animation.animation.rainbowspace.RainbowSpace* *method*), 32  
`after_draw()` (*adafruit\_led\_animation.animation.sparkle.Sparkle* *method*), 29  
`after_draw()` (*adafruit\_led\_animation.animation.sparklepulse.SparklePulse* *method*), 32  
`AMBER` (*in module adafruit\_led\_animation.color*), 15  
`animate()` (*adafruit\_led\_animation.animation.Animation* *method*), 14  
`animate()` (*adafruit\_led\_animation.group.AnimationGroup* *method*), 20  
`animate()` (*adafruit\_led\_animation.sequence.AnimateOnce* *method*), 21  
`animate()` (*adafruit\_led\_animation.sequence.AnimationSequence* *method*), 23  
`AnimateOnce` (*class in adafruit\_led\_animation.sequence*), 21  
`Animation` (*class in adafruit\_led\_animation.animation*), 14  
`AnimationGroup` (*class in adafruit\_led\_animation.group*), 20  
`AnimationSequence` (*class in adafruit\_led\_animation.sequence*), 22  
`AQUA` (*in module adafruit\_led\_animation.color*), 15  
`auto_write` (*adafruit\_led\_animation.helper.PixelMap* *attribute*), 18

## B

`bar_color()` (*adafruit\_led\_animation.animation.chase.Chase* *method*), 26  
`bar_color()` (*adafruit\_led\_animation.animation.rainbowchase.RainbowChase* *method*), 30  
`BLACK` (*in module adafruit\_led\_animation.color*), 15

Blink (*class in adafruit\_led\_animation.animation.blink*), 23

BLUE (*in module adafruit\_led\_animation.color*), 15

brightness (*adafruit\_led\_animation.helper.PixelMap attribute*), 18

## C

calculate\_intensity() (*in module adafruit\_led\_animation.color*), 16

Chase (*class in adafruit\_led\_animation.animation.chase*), 25

color (*adafruit\_led\_animation.animation.Animation attribute*), 14

color (*adafruit\_led\_animation.group.AnimationGroup attribute*), 20

color (*adafruit\_led\_animation.sequence.AnimationSequence attribute*), 23

ColorCycle (*class in adafruit\_led\_animation.animation.colorcycle*), 24

colorwheel() (*in module adafruit\_led\_animation.color*), 16

Comet (*class in adafruit\_led\_animation.animation.comet*), 26

current\_animation (*adafruit\_led\_animation.sequence.AnimationSequence attribute*), 23

CYAN (*in module adafruit\_led\_animation.color*), 15

cycle\_count (*adafruit\_led\_animation.animation.Animation attribute*), 14

cycle\_count (*adafruit\_led\_animation.group.AnimationGroup attribute*), 20

## D

draw() (*adafruit\_led\_animation.animation.Animation method*), 14

draw() (*adafruit\_led\_animation.animation.chase.Chase method*), 26

draw() (*adafruit\_led\_animation.animation.colorcycle.ColorCycle method*), 25

draw() (*adafruit\_led\_animation.animation.comet.Comet method*), 27

draw() (*adafruit\_led\_animation.animation.pulse.Pulse method*), 27

draw() (*adafruit\_led\_animation.animation.rainbow.Rainbow method*), 28

draw() (*adafruit\_led\_animation.animation.sparkle.Sparkle method*), 29

draw() (*adafruit\_led\_animation.animation.sparklepulse.SparklePulse method*), 32

draw\_count (*adafruit\_led\_animation.animation.Animation attribute*), 14

draw\_count (*adafruit\_led\_animation.group.AnimationGroup attribute*), 20

## F

fill() (*adafruit\_led\_animation.animation.Animation method*), 14

fill() (*adafruit\_led\_animation.group.AnimationGroup method*), 20

fill() (*adafruit\_led\_animation.helper.PixelMap method*), 18

fill() (*adafruit\_led\_animation.sequence.AnimationSequence method*), 23

freeze() (*adafruit\_led\_animation.animation.Animation method*), 14

freeze() (*adafruit\_led\_animation.group.AnimationGroup method*), 20

freeze() (*adafruit\_led\_animation.sequence.AnimationSequence method*), 23

## G

generate\_rainbow() (*adafruit\_led\_animation.animation.rainbow.Rainbow method*), 28

generate\_rainbow() (*adafruit\_led\_animation.animation.rainbowsparkle.RainbowSparkle method*), 32

GOLD (*in module adafruit\_led\_animation.color*), 15

GREEN (*in module adafruit\_led\_animation.color*), 15

## H

horizontal\_lines() (*adafruit\_led\_animation.helper.PixelMap class method*), 18

horizontal\_strip\_gridmap() (*in module adafruit\_led\_animation.helper*), 19

## J

JADE (*in module adafruit\_led\_animation.color*), 15

## M

MAGENTA (*in module adafruit\_led\_animation.color*), 16

## N

next() (*adafruit\_led\_animation.sequence.AnimationSequence method*), 23

notify\_cycles (*adafruit\_led\_animation.animation.Animation attribute*), 14

notify\_cycles (*adafruit\_led\_animation.group.AnimationGroup attribute*), 20

## O

OLD\_LACE (*in module adafruit\_led\_animation.color*), 16

on\_cycle\_complete() (*adafruit\_led\_animation.animation.Animation method*), 15

`on_cycle_complete()` (`adafruit_led_animation.animation.rainbowchase.RainbowChase` method), 30  
`on_cycle_complete()` (`adafruit_led_animation.group.AnimationGroup` method), 20  
`on_cycle_complete()` (`adafruit_led_animation.sequence.AnimateOnce` method), 22  
`on_cycle_complete()` (`adafruit_led_animation.sequence.AnimationSequence` method), 23  
 ORANGE (in module `adafruit_led_animation.color`), 16

## P

`peers` (`adafruit_led_animation.animation.Animation` attribute), 15  
 PINK (in module `adafruit_led_animation.color`), 16  
`PixelMap` (class in `adafruit_led_animation.helper`), 17  
`PixelSubset` (class in `adafruit_led_animation.helper`), 19  
`Pulse` (class in `adafruit_led_animation.animation.pulse`), 27  
`pulse_generator()` (in module `adafruit_led_animation.helper`), 19  
 PURPLE (in module `adafruit_led_animation.color`), 16

## R

`Rainbow` (class in `adafruit_led_animation.animation.rainbow`), 28  
 RAINBOW (in module `adafruit_led_animation.color`), 16  
`RainbowChase` (class in `adafruit_led_animation.animation.rainbowchase`), 29  
`RainbowComet` (class in `adafruit_led_animation.animation.rainbowcomet`), 30  
`RainbowSparkle` (class in `adafruit_led_animation.animation.rainbowsparkle`), 31  
`random()` (`adafruit_led_animation.sequence.AnimationSequence` method), 23  
 RED (in module `adafruit_led_animation.color`), 16  
`reset()` (`adafruit_led_animation.animation.Animation` method), 15  
`reset()` (`adafruit_led_animation.animation.chase.Chase` method), 26  
`reset()` (`adafruit_led_animation.animation.colorcycle.ColorCycle` method), 25  
`reset()` (`adafruit_led_animation.animation.comet.Comet` method), 27  
`reset()` (`adafruit_led_animation.animation.pulse.Pulse` method), 27  
`reset()` (`adafruit_led_animation.animation.rainbow.RainbowChase` method), 28  
`reset()` (`adafruit_led_animation.group.AnimationGroup` method), 21  
`reset()` (`adafruit_led_animation.sequence.AnimationSequence` method), 23  
`resume()` (`adafruit_led_animation.animation.Animation` method), 15  
`resume()` (`adafruit_led_animation.group.AnimationGroup` method), 21  
`resume()` (`adafruit_led_animation.sequence.AnimationSequence` method), 23  
`reverse()` (`adafruit_led_animation.animation.chase.Chase` attribute), 26  
 RGBW\_WHITE\_RGB (in module `adafruit_led_animation.color`), 16  
 RGBW\_WHITE\_RGBW (in module `adafruit_led_animation.color`), 16  
 RGBW\_WHITE\_W (in module `adafruit_led_animation.color`), 16

## S

`show()` (`adafruit_led_animation.animation.Animation` method), 15  
`show()` (`adafruit_led_animation.group.AnimationGroup` method), 21  
`show()` (`adafruit_led_animation.helper.PixelMap` method), 18  
`show()` (`adafruit_led_animation.sequence.AnimationSequence` method), 23  
`Solid` (class in `adafruit_led_animation.animation.solid`), 24  
`space_color()` (`adafruit_led_animation.animation.chase.Chase` method), 26  
`Sparkle` (class in `adafruit_led_animation.animation.sparkle`), 29  
`SparklePulse` (class in `adafruit_led_animation.animation.sparklepulse`), 32  
`speed` (`adafruit_led_animation.animation.Animation` attribute), 15

## T

TEAL (in module `adafruit_led_animation.color`), 16

## V

`vertical_lines()` (`adafruit_led_animation.helper.PixelMap` class method), 18  
`vertical_strip_gridmap()` (in module `adafruit_led_animation.helper`), 19

## W

WHITE (in module `adafruit_led_animation.color`), 16

## Y

YELLOW (*in module `adafruit_led_animation.color`*), [16](#)