Adafruitturtle Library Documentation

Release 1.0

Adafruit

Jun 07, 2021

Contents

Turtle graphics library for CircuitPython and displayio

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython
- Bus Device

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-turtle
```

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

```
sudo pip3 install adafruit-circuitpython-turtle
```

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-turtle
```

CHAPTER 3

Usage Example

```
import board
from adafruit_turtle import Color, turtle

turtle = turtle(board.DISPLAY)
starsize = min(board.DISPLAY.width, board.DISPLAY.height) * 0.9  # 90% of screensize

print("Turtle time! Lets draw a star")

turtle.pencolor(Color.BLUE)

turtle.penup()
turtle.goto(-starsize/2, 0)
turtle.pendown()

start = turtle.pos()
while True:
    turtle.forward(starsize)
    turtle.left(170)
    if abs(turtle.pos() - start) < 1:
        break

while True:
    pass
```

CHAPTER 4

Contributing

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

CHAPTER 5

Documentation

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

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/turtle_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import board
from adafruit_turtle import Color, turtle

turtle = turtle(board.DISPLAY)
starsize = min(board.DISPLAY.width, board.DISPLAY.height) * 0.9  # 90% of screensize

print("Turtle time! Lets draw a star")

turtle.pencolor(Color.BLUE)
turtle.setheading(90)

turtle.penup()
turtle.goto(-starsize / 2, 0)
turtle.pendown()

start = turtle.pos()
while True:
    turtle.forward(starsize)
    turtle.left(170)
    if abs(turtle.pos() - start) < 1:
        break

while True:
    pass
```

6.2 adafruit_turtle

- Originals Author(s): LadyAda and Dave Astels

6.2.1 Implementation Notes

Hardware:

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

class adafruit_turtle.**Color**
Standard colors

class adafruit_turtle.**Vec2D**(*x*, *y*)
A 2 dimensional vector class, used as a helper class for implementing turtle graphics. May be useful for turtle graphics programs also. Derived from tuple, so a vector is a tuple!

rotate(*angle*)
Rotate self counterclockwise by angle.

Parameters **angle** – how much to rotate

class adafruit_turtle.**turtle**(*display=None*, *scale=1*)
A Turtle that can be given commands to draw.

back(*distance*)
Move the turtle backward by distance, opposite to the direction the turtle is headed. Does not change the turtle's heading.

Parameters **distance** – how far to move (integer or float)

backward(*distance*)
Move the turtle backward by distance, opposite to the direction the turtle is headed. Does not change the turtle's heading.

Parameters **distance** – how far to move (integer or float)

bgcolor(*c=None*)
Return or set the background color.

bgcolor() Return the current backgroud color as color specification string. May be used as input to another color/ pencolor/fillcolor call.

bgcolor(colorvalue) Set backgroud color to colorvalue, which is a 24-bit integer such as 0xFF0000. The Color class provides the available values: WHITE, BLACK, RED, ORANGE, YELLOW, GREEN, BLUE, PURPLE, PINK

bgpic(*picname=None*)
Set background image or return name of current backgroundimage. Optional argument: picname – a string, name of an image file or "nopic". If picname is a filename, set the corresponding image as background. If picname is "nopic", delete backgroundimage, if present. If picname is None, return the filename of the current backgroundimage.

bk(*distance*)
Move the turtle backward by distance, opposite to the direction the turtle is headed. Does not change the turtle's heading.

Parameters **distance** – how far to move (integer or float)

changeturtle(*source=None*, *dimensions=(12*, *12)*)
Change the turtle. if a string is provided, its a path to an image opened via OnDiskBitmap if a tilegrid is provided, it replace the default one for the turtle shape. if no argument is provided, the default shape will be restored

circle(*radius*, *extent=None*, *steps=None*)
Draw a circle with given radius. The center is radius units left of the turtle; extent - an angle - determines which part of the circle is drawn. If extent is not given, draw the entire circle. If extent is not a full circle, one endpoint of the arc is the current pen position. Draw the arc in counterclockwise direction if radius is positive, otherwise in clockwise direction. Finally the direction of the turtle is changed by the amount of extent.

As the circle is approximated by an inscribed regular polygon, steps determines the number of steps to use. If not given, it will be calculated automatically. May be used to draw regular polygons.

Parameters

- **radius** – the radius of the circle
- **extent** – the arc of the circle to be drawn
- **steps** – how many points along the arc are computed

clear()
Delete the turtle's drawings from the screen. Do not move turtle.

clearstamp(*stampid*)
Delete stamp with given stampid.

Parameters **stampid** – the id of the stamp to be deleted

clearstamps(*n=None*)
Delete all or first/last n of turtle's stamps. If n is None, delete all stamps, if n > 0 delete first n stamps, else if n < 0 delete last n stamps.

Parameters **n** – how many stamps to delete (None means delete them all)

degrees(*fullcircle=360*)
Set angle measurement units, i.e. set number of "degrees" for a full circle. Default value is 360 degrees.

Parameters **fullcircle** – the number of degrees in a full circle

distance(*x1*, *y1=None*)
Return the distance from the turtle to (x,y) or the vector, in turtle step units.

Parameters

- **x** – a number or a pair/vector of numbers
- **y** – a number if x is a number, else None

dot(*size=None*, *color=None*)
Draw a circular dot with diameter size, using color. If size is not given, the maximum of pensize+4 and 2*pensize is used.

Parameters

- **size** – the diameter of the dot
- **color** – the color of the dot

down()
Pull the pen down - drawing when moving.

fd(*distance*)
Move the turtle forward by the specified distance, in the direction the turtle is headed.

Parameters **distance** – how far to move (integer or float)

forward(*distance*)
Move the turtle forward by the specified distance, in the direction the turtle is headed.

Parameters **distance** – how far to move (integer or float)

goto(*x1*, *y1=None*)
If y1 is None, x1 must be a pair of coordinates or an (x, y) tuple

Move turtle to an absolute position. If the pen is down, draw line. Does not change the turtle's orientation.

Parameters

- **x1** – a number or a pair of numbers
- **y1** – a number or None

heading()
Return the turtle's current heading (value depends on the turtle mode, see mode()).

hideturtle()
Make the turtle invisible.

home()
Move turtle to the origin - coordinates (0,0) - and set its heading to its start-orientation (which depends on the mode, see mode()).

ht()
Make the turtle invisible.

isdown()
Return True if pen is down, False if it's up.

isvisible()
Return True if the Turtle is shown, False if it's hidden.

left(*angle*)
Turn turtle left by angle units. (Units are by default degrees, but can be set via the degrees() and radians() functions.) Angle orientation depends on the turtle mode, see mode().

Parameters **angle** – how much to rotate to the left (integer or float)

lt(*angle*)
Turn turtle left by angle units. (Units are by default degrees, but can be set via the degrees() and radians() functions.) Angle orientation depends on the turtle mode, see mode().

Parameters **angle** – how much to rotate to the left (integer or float)

mode(*mode=None*)
Set turtle mode ("standard" or "logo") and perform reset. If mode is not given, current mode is returned.

Mode "standard" is compatible with old turtle. Mode "logo" is compatible with most Logo turtle graphics.

Parameters **mode** – one of the strings "standard" or "logo"

pd()
Pull the pen down - drawing when moving.

pencolor(*c=None*)
Return or set the pencolor.

pencolor() Return the current pencolor as color specification string or as a tuple (see example). May be used as input to another color/ pencolor/fillcolor call.

pencolor(colorvalue) Set pencolor to colorvalue, which is a 24-bit integer such as 0xFF0000. The Color class provides the available values: BLACK, WHITE, RED, YELLOW, ORANGE, GREEN, BLUE, PURPLE, PINK GRAY, LIGHT_GRAY, BROWN, DARK_GREEN, TURQUOISE, DARK_BLUE, DARK_RED

pendown()
Pull the pen down - drawing when moving.

pensize(*width=None*)
Set the line thickness to width or return it. If no argument is given, the current pensize is returned.

Parameters **width** –

- a positive number

penup()
Pull the pen up - no drawing when moving.

pos()
Return the turtle's current location (x,y) (as a Vec2D vector).

position()
Return the turtle's current location (x,y) (as a Vec2D vector).

pu()
Pull the pen up - no drawing when moving.

radians()
Set the angle measurement units to radians. Equivalent to degrees(2*math.pi).

reset()
Delete the turtle's drawings from the screen, re-center the turtle and set variables to the default values.

right(*angle*)
Turn turtle right by angle units. (Units are by default degrees, but can be set via the degrees() and radians() functions.) Angle orientation depends on the turtle mode, see mode().

Parameters **angle** – how much to rotate to the right (integer or float)

rt(*angle*)
Turn turtle right by angle units. (Units are by default degrees, but can be set via the degrees() and radians() functions.) Angle orientation depends on the turtle mode, see mode().

Parameters **angle** – how much to rotate to the right (integer or float)

seth(*to_angle*)
Set the orientation of the turtle to to_angle. Here are some common directions in degrees:

standard mode | logo mode 0 - east | 0 - north 90 - north | 90 - east 180 - west | 180 - south 270 - south | 270 - west

Parameters **to_angle** – the new turtle heading

setheading(*to_angle*)
Set the orientation of the turtle to to_angle. Here are some common directions in degrees:

standard mode | logo mode 0 - east | 0 - north 90 - north | 90 - east 180 - west | 180 - south 270 - south | 270 - west

Parameters **to_angle** – the new turtle heading

setpos(*x1*, *y1=None*)
If y1 is None, x1 must be a pair of coordinates or an (x, y) tuple

Move turtle to an absolute position. If the pen is down, draw line. Does not change the turtle's orientation.

Parameters

- **x1** – a number or a pair of numbers
- **y1** – a number or None

setposition(*x1*, *y1=None*)
If y1 is None, x1 must be a pair of coordinates or an (x, y) tuple

Move turtle to an absolute position. If the pen is down, draw line. Does not change the turtle's orientation.

Parameters

- **x1** – a number or a pair of numbers
- **y1** – a number or None

setx(*x*)
Set the turtle's first coordinate to x, leave second coordinate unchanged.

Parameters **x** – new value of the turtle's x coordinate (a number)

sety(*y*)
Set the turtle's second coordinate to y, leave first coordinate unchanged.

Parameters **y** – new value of the turtle's y coordinate (a number)

showturtle()
Make the turtle visible.

speed(*speed=None*)
Set the turtle's speed to an integer value in the range 0..10. If no argument is given, return current speed.

If input is a number greater than 10 or smaller than 1, speed is set to 0. Speedstrings are mapped to speedvalues as follows:

"fastest": 0 "fast": 10 "normal": 6 "slow": 3 "slowest": 1 Speeds from 1 to 10 enforce increasingly faster animation of line drawing and turtle turning.

Attention: speed = 0 means that no animation takes place. forward/back makes turtle jump and likewise left/right make the turtle turn instantly.

Parameters **speed** – the new turtle speed (0..10) or None

st()
Make the turtle visible.

stamp(*bitmap=None*, *palette=None*)
Stamp a copy of the turtle shape onto the canvas at the current turtle position. Return a stamp_id for that stamp, which can be used to delete it by calling clearstamp(stamp_id).

towards(*x1*, *y1=None*)
Return the angle between the line from turtle position to position specified by (x,y) or the vector. This depends on the turtle's start orientation which depends on the mode - "standard" or "logo").

Parameters

- **x** – a number or a pair/vector of numbers
- **y** – a number if x is a number, else None

up()
Pull the pen up - no drawing when moving.

width(*width=None*)
Set the line thickness to width or return it. If no argument is given, the current pensize is returned.

Parameters **width** –

- a positive number

window_height()
Return the height of the turtle window.

window_width()
Return the width of the turtle window.

xcor()
Return the turtle's x coordinate.

ycor()
Return the turtle's y coordinate.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

U

V

W

X

Y