
Stage Documentation

Release 1.0

Radomir Dopieralski

Aug 26, 2022

Contents

1	API Reference	3
1.1	stage	3
1.2	ugame	4
	Python Module Index	7
	Index	9

Stage is a library that lets you display tile grids and sprites on SPI-based RGB displays in CircuitPython. It is mostly made with video games in mind, but it can be useful in making any kind of graphical user interface too.

For performance reasons, a part of this library has been written in C and has to be compiled as part of the CircuitPython firmware as the `_stage` module. For memory saving reasons, it's best if this library is also included in the firmware, as a frozen module.

The API reference is available at <http://circuitpython-stage.readthedocs.io>.

1.1 stage

class `stage.BMP16` (*filename*)
Read 16-color BMP files.

class `stage.Bank` (*buffer=None, palette=None*)
Store graphics for the tiles and sprites.

A single bank stores exactly 16 tiles, each 16x16 pixels in 16 possible colors, and a 16-color palette. We just like the number 16.

classmethod `from_bmp16` (*filename*)
Read the bank from a BMP file.

classmethod `from_image` (*filename*)
Read the bank from an image file.

class `stage.Grid` (*bank, width=8, height=8, palette=None, buffer=None*)
A grid is a layer of tiles that can be displayed on the screen. Each square can contain any of the 16 tiles from the associated bank.

move (*x, y, z=None*)
Shift the whole layer respective to the screen.

tile (*x, y, tile=None*)
Get or set what tile is displayed in the given place.

class `stage.PNG16` (*filename*)
Read 16-color PNG files.

class `stage.Sprite` (*bank, frame, x, y, z=0, rotation=0, palette=None*)
A sprite is a layer containing just a single tile from the associated bank, that can be positioned anywhere on the screen.

move (*x, y, z=None*)

Move the sprite to the given place.

set_frame (*frame=None, rotation=None*)

Set the current graphic and rotation of the sprite.

The possible values for rotation are: 0 - none, 1 - 90 degrees clockwise, 2 - 180 degrees, 3 - 90 degrees counter-clockwise, 4 - mirrored, 5 - 90 degrees clockwise and mirrored, 6 - 180 degrees and mirrored, 7 - 90 degrees counter-clockwise and mirrored.

class `stage.Stage` (*display, fps=6, scale=None*)

Represents what is being displayed on the screen.

The `display` parameter is `displayio.Display` representing an initialized display connected to the device.

The `fps` specifies the maximum frame rate to be enforced.

The `scale` specifies an optional scaling up of the display, to use 2x2 or 3x3, etc. pixels. If not specified, it is inferred from the display size (displays wider than 256 pixels will have `scale=2`, for example).

render_block (*x0=None, y0=None, x1=None, y1=None*)

Update a rectangle of the screen.

render_sprites (*sprites*)

Update the spots taken by all the sprites in the list.

tick ()

Wait for the start of the next frame.

class `stage.Text` (*width, height, font=None, palette=None, buffer=None*)

Text layer. For displaying text.

char (*x, y, c=None, highlight=False*)

Get or set the character at the given location.

clear ()

Clear all text from the layer.

cursor (*x=None, y=None*)

Move the text cursor to the specified row and column.

move (*x, y, z=None*)

Shift the whole layer respective to the screen.

text (*text, highlight=False*)

Display text starting at the current cursor location. Return the dimensions of the rendered text.

class `stage.WallGrid` (*grid, walls, bank, palette=None*)

A special grid, shifted from its parents by half a tile, useful for making nice-looking corners of walls and similar structures.

`stage.collide` (*ax0, ay0, ax1, ay1, bx0, by0, bx1=None, by1=None*)

Return True if the two rectangles intersect.

`stage.color565` (*r, g, b*)

Convert 24-bit RGB color to 16-bit.

1.2 ugame

`ugame.display`

An initialized display, that can be used for creating Stage objects.

ugame.**buttons**

An instance of `GamePad` or other similar class, that has a `get_pressed` method for retrieving a bit mask of pressed buttons. That value can be then checked with `&` operator against the constants: `K_UP`, `K_DOWN`, `K_LEFT`, `K_RIGHT`, `K_X`, `K_O` and on some platforms also: `K_START` and `K_SELECT`.

ugame.**audio**

And instance of the `Audio` or other similar class, that has `play`, `stop` and `mute` methods.

S

stage, 3

U

ugame, 4

A

audio (*in module ugame*), 5

B

Bank (*class in stage*), 3

BMP16 (*class in stage*), 3

buttons (*in module ugame*), 4

C

char () (*stage.Text method*), 4

clear () (*stage.Text method*), 4

collide () (*in module stage*), 4

color565 () (*in module stage*), 4

cursor () (*stage.Text method*), 4

D

display (*in module ugame*), 4

F

from_bmp16 () (*stage.Bank class method*), 3

from_image () (*stage.Bank class method*), 3

G

Grid (*class in stage*), 3

M

move () (*stage.Grid method*), 3

move () (*stage.Sprite method*), 3

move () (*stage.Text method*), 4

P

PNG16 (*class in stage*), 3

R

render_block () (*stage.Stage method*), 4

render_sprites () (*stage.Stage method*), 4

S

set_frame () (*stage.Sprite method*), 4

Sprite (*class in stage*), 3

Stage (*class in stage*), 4

stage (*module*), 3

T

Text (*class in stage*), 4

text () (*stage.Text method*), 4

tick () (*stage.Stage method*), 4

tile () (*stage.Grid method*), 3

U

ugame (*module*), 4

W

WallGrid (*class in stage*), 4