Music Editor:

The mode to edit musics in which the sounds are arranged in order of playback.

Other resource creation methods

Pyxel images and tilemaps can also be created in the following way:

- Create an image from a list of strings with `Image.set` or `Tilemap.set` function
- Load a png file in Pyxel palette with `Image.load` function

Pyxel sounds can also be created in the following way:

- Create a sound from strings with `Sound.set` or `Music.set` function

Please refer to the API reference for usage of these functions.

How to Create a Stand-Alone Executable

By using the attached Pyxel Packager, a stand-alone executable that will work even in environments where Python is not installed can be created.

To create a stand-alone executable, specify the Python file to be used to launch the application with the `pyxelpackager` command as follows:

```
pyxelpackager python_file
```

When the process is complete, a stand-alone executable is created in the `dist` folder.

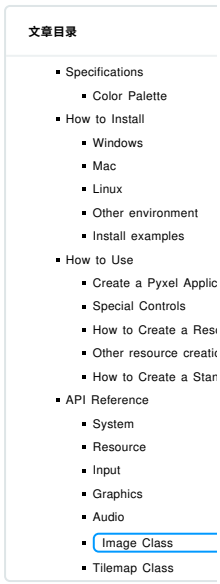
If resources such as .pyxres and .png files are also necessary, put them under the `assets` folder and they will be included.

It is also possible to specify an icon with the `-i icon_file` option.

API Reference

System

- `width`, `height`
  The width and height of the screen
- `frame_count`
  The number of the elapsed frames
- `init(width, height, [caption], [scale], [palette],`



`[fps], [border_width], [border_color], [quit_key])`
Initialize the Pyxel application with screen size (`width` , `height`). The maximum width and height of the screen is 256 It is also possible to specify the window title with `caption` , the display magnification with `scale` , the palette color with `palette` , the frame rate with `fps` , the margin width and color outside the screen with `border_width` and `border_color` , and the key to quit the application with `quit_key` . `palette` is specified as a list of 16 elements of 24 bit color, `border_color` as 24 bit color.
e.g. `pyxel.init(160, 120, caption="Pyxel with PICO-8 palette", palette=[0x000000, 0x1D2B53, 0x7E2553, 0x008751, 0xAB5236, 0x5F574F, 0xC2C3C7, 0xFFF1E8, 0xFF004D, 0xFFA300, 0xFFEC27, 0x00E436, 0x29ADFF, 0x83769C, 0xFF77A8, 0xFFCCAA], quit_key=pyxel.KEY_NONE)`

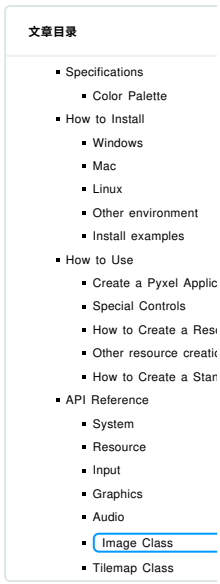
- `run(update, draw)`
  Start the Pyxel application and call `update` function for frame update and `draw` function for drawing
- `quit()`
  Quit the Pyxel application at the end of the current frame
- `flip()`
  Force drawing the screen (do not use in normal applications)
- `show()`
  Draw the screen and wait forever (do not use in normal applications)

Resource

- `save(filename)`
  Save the resource file (.pyxres) to the directory of the execution script
- `load(filename, [image], [tilemap], [sound], [music])`
  Read the resource file (.pyxres) from the directory of the execution script. If `False` is specified for the resource type (image/tilemap/sound/music), the resource will not be loaded.

Input

- `mouse_x` , `mouse_y`
  The current position of the mouse cursor
- `btn(key)`
  Return `True` if `key` is pressed, otherwise return `False` (key definition list)
- `btnp(key, [hold], [period])`



Return `True` if `key` is pressed at that frame, otherwise return `False`. When `hold` and `period` are specified, `True` will be returned at the `period` frame interval when the `key` is held down for more than `hold` frames

- `btnr(key)`
  Return `True` if `key` is released at that frame, otherwise return `False`

- `mouse(visible)`
  If `visible` is `True`, show the mouse cursor. If `False`, hide it. Even if the mouse cursor is not displayed, its position is updated.

Graphics

- `image(img, [system])`
  Operate the image bank `img` (0-2) (see the Image class). If `system` is `True`, the image bank for system can be accessed. 3 is for the font and resource editor. 4 is for the display screen
  e.g. `pyxel.image(0).load(0, 0, "title.png")`

- `tilemap(tm)`
  Operate the tilemap `tm` (0-7) (see the Tilemap class)

- `clip(x, y, w, h)`
  Set the drawing area of the screen from (`x`, `y`) to width `w` and height `h`. Reset the drawing area to full screen with `clip()`

- `pal(col1, col2)`
  Replace color `col1` with `col2` at drawing. `pal()` to reset to the initial palette

- `cls(col)`
  Clear screen with color `col`

- `pget(x, y)`
  Get the color of the pixel at (`x`, `y`)

- `pset(x, y, col)`
  Draw a pixel of color `col` at (`x`, `y`)

- `line(x1, y1, x2, y2, col)`
  Draw a line of color `col` from (`x1`, `y1`) to (`x2`, `y2`)

- `rect(x, y, w, h, col)`
  Draw a rectangle of width `w`, height `h` and color `col` from (`x`, `y`)

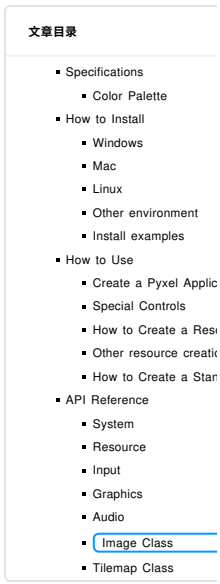
- `rectb(x, y, w, h, col)`
  Draw the outline of a rectangle of width `w`, height `h` and

color `col` from (`x` , `y`)

- `circ(x, y, r, col)`
  Draw a circle of radius `r` and color `col` at (`x` , `y`)
- `circb(x, y, r, col)`
  Draw the outline of a circle of radius `r` and color `col` at (`x` , `y`)
- `tri(x1, y1, x2, y2, x3, y3, col)`
  Draw a triangle with vertices (`x1` , `y1`), (`x2` , `y2`), (`x3` , `y3`) and color `col`
- `trib(x1, y1, x2, y2, x3, y3, col)`
  Draw the outline of a triangle with vertices (`x1` , `y1`), (`x2` , `y2`), (`x3` , `y3`) and color `col`
- `blt(x, y, img, u, v, w, h, [colkey])`
  Copy the region of size (`w` , `h`) from (`u` , `v`) of the image bank `img` (0-2) to (`x` , `y`). If negative value is set for `w` and/or `h` , it will reverse horizontally and/or vertically. If `colkey` is specified, treated as transparent color
- `bltm(x, y, tm, u, v, w, h, [colkey])`
  Draw the tilemap `tm` (0-7) to (`x` , `y`) according to the tile information of size (`w` , `h`) from (`u` , `v`). If `colkey` is specified, treated as transparent color. A tile of the tilemap is drawn with a size of 8x8, and if the tile number is 0, indicates the region (0, 0)-(7, 7) of the image bank, if 1, indicates (8, 0)-(15, 0)
- `text(x, y, s, col)`
  Draw a string `s` of color `col` at (`x` , `y`)

Audio

- `sound(snd, [system])`
  Operate the sound `snd` (0-63) (see the Sound class). If `system` is `True` , the sound 64 for system can be accessed
  e.g. `pyxel.sound(0).speed = 60`
- `music(msc)`
  Operate the music `msc` (0-7) (see the Music class)
- `play_pos(ch)`
  Get the sound playback position of channel `ch` . The 100's and 1000's indicate the sound number and the 1's and 10's indicate the note number. When playback is stopped, return `-1`
- `play(ch, snd, loop=False)`
  Play the sound `snd` (0-63) on channel `ch` (0-3). Play in order when `snd` is a list



- `playm(msc, loop=False)`
  Play the music `msc` (0-7)

- `stop([ch])`
  Stop playback of all channels. If `ch` (0-3) is specified, stop the corresponding channel only

Image Class

- `width`, `height`
  The width and height of the image

- `data`
  The data of the image (256x256 two-dimentional list)

- `get(x, y)`
  Retrieve the data of the image at (`x`, `y`)

- `set(x, y, data)`
  Set the data of the image at (`x`, `y`) by a value or a list of strings
  e.g. `pyxel.image(0).set(10, 10, ["1234", "5678", "9abc", "defg"])`

- `load(x, y, filename)`
  Read the png image from the directory of the execution script at (`x`, `y`)

- `copy(x, y, img, u, v, w, h)`
  Copy the region of size (`w`, `h`) from (`u`, `v`) of the image bank `img` (0-2) to (`x`, `y`)

Tilemap Class

- `width`, `height`
  The width and height of the tilemap

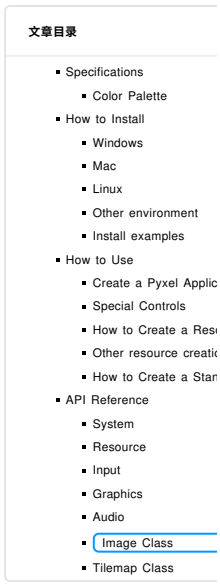
- `data`
  The data of the tilemap (256x256 two-dimentional list)

- `refimg`
  The image bank referenced by the tilemap

- `get(x, y)`
  Retrieve the data of the tilemap at (`x`, `y`)

- `set(x, y, data)`
  Set the data of the tilemap at (`x`, `y`) by a value or a list of strings.
  e.g. `pyxel.tilemap(0).set(0, 0, ["000102", "202122", "a0a1a2", "b0b1b2"])`

- `copy(x, y, tm, u, v, w, h)`



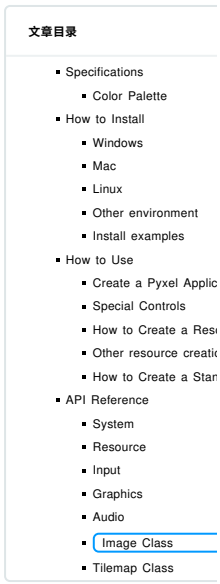
Copy the region of size (`w` , `h`) from (`u` , `v`) of the tilemap `tm` (0-7) to (`x` , `y`)

Sound Class

- `note`
  List of note(0-127) (33 = 'A2' = 440Hz)
- `tone`
  List of tone(0:Triangle / 1:Square / 2:Pulse / 3:Noise)
- `volume`
  List of volume(0-7)
- `effect`
  List of effects(0:None / 1:Slide / 2:Vibrato / 3:FadeOut)
- `speed`
  The length of one note(120 = 1 second per tone)
- `set(note, tone, volume, effect, speed)`
  Set a note, tone, volume, and effect with a string. If the tone, volume, and effect length are shorter than the note, it is repeated from the beginning
- `set_note(note)`
  Set the note with a string made of 'CDEFGAB'+'#-'+'0123' or 'R'. Case-insensitive and whitespace is ignored
  e.g. `pyxel.sound(0).set_note("G2B-2D3R RF3F3F3")`
- `set_tone(tone)`
  Set the tone with a string made of 'TSPN'. Case-insensitive and whitespace is ignored
  e.g. `pyxel.sound(0).set_tone("TTSS PPPN")`
- `set_volume(volume)`
  Set the volume with a string made of '01234567'. Case-insensitive and whitespace is ignored
  e.g. `pyxel.sound(0).set_volume("7777 7531")`
- `set_effect(effect)`
  Set the effect with a string made of 'NSVF'. Case-insensitive and whitespace is ignored
  e.g. `pyxel.sound(0).set_effect("NFNF NVVS")`

Music Class

- `ch0`
  List of sound(0-63) play on channel 0. If an empty list is specified, the channel is not used for playback
- `ch1`
  List of sound(0-63) play on channel 1. If an empty list is



specified, the channel is not used for playback

- `ch2`
  List of sound(0-63) play on channel 2. If an empty list is specified, the channel is not used for playback
- `ch3`
  List of sound(0-63) play on channel 3. If an empty list is specified, the channel is not used for playback
- `set(ch0, ch1, ch2, ch3)`
  Set the list of sound(0-63) of all channels. If an empty list is specified, that channel is not used for playback
  e.g. `pyxel.music(0).set([0, 1], [2, 3], [4], [])`
- `set_ch0(data)`
  Set the list of sound(0-63) of channel 0
- `set_ch1(data)`
  Set the list of sound(0-63) of channel 1
- `set_ch2(data)`
  Set the list of sound(0-63) of channel 2
- `set_ch3(data)`
  Set the list of sound(0-63) of channel 3

How to Contribute

Submitting an issue

Use the issue tracker to submit bug reports and feature/enhancement requests. Before submitting a new issue, search the issue tracker to ensure that there is no similar open issue.

When submitting a report, select the appropriate template from this link.

Manual testing

Anyone manually testing the code and reporting bugs or suggestions for enhancements in the issue tracker are very welcome!

Submitting a pull request

Patches/fixes are accepted in form of pull requests (PRs). Make sure the issue the pull request addresses is open in the issue tracker.

Submitted pull request is deemed to have agreed to publish under MIT license.

Other Information