SimpleGUICS2Pygame

Jan 12, 2021

Contents

It is primarily a standard Python (**2 and 3**) module reimplementing the SimpleGUI particular module of CodeSkulptor and CodeSkulptor3 (a Python browser environment). This is in fact a package also with other modules adapted from CodeSkulptor.

Simply change

```
import simplegui
```

by

```
try:
    import simplegui
except ImportError:
    import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
```

in your CodeSkulptor program and your program **run both** in CodeSkulptor and *standard Python* with this module (and Pygame).

Online HTML documentation on **Read The Docs**. (You can also see the online SimpleGUI documentation on CodeSkulptor or SimpleGUI documentation on CodeSkulptor3.)

Sources and installers on Bitbucket: https://bitbucket.org/OPiMedia/simpleguics2pygame
and on **PyPI**: https://pypi.org/project/SimpleGUICS2Pygame/ .

CHAPTER 1

If you have some problem

First, read this short main documentation page, this Compatibility page and this Tips page.

If you have problem with some command, you can see its documentation in the modules page or by the genindex page .

Next, you can search in Stack Overflow. If you don't find answer, you can ask question like this.

Finally you can email me. I will try to help you with pleasure. (You can write me in French.)

CHAPTER 2

Installation

2.1 Before the installation of SimpleGUICS2Pygame

Obviously you need Python.

If pip (the Python Package Installer) is not installed on your system, then install it with:

```
$ python -m ensurepip --user
```

Note that `$` represents the prompt and do *not* be entered by you.

If several Python implementations are installed, maybe you must use something like `python2` or `python3` instead `python` command.

With the `--user` option the installation is made in the user directory and doesn't require administrator rights.

Before installation, as a precaution, upgrade necessary installation packages:

```
$ python -m pip install pip setuptools wheel --user --upgrade
```

2.2 Installation of SimpleGUICS2Pygame

```
$ python -m pip install SimpleGUICS2Pygame --user --upgrade
```

Followed requirements are automatically installed.

But if the installation failed, then install them separately and after that try again to install SimpleGUICS2Pygame.

On Arch Linux you can use this package installation script (written by Danny Fajardo): Arch_Linux/PKGBUILD.

2.3 Package pygame required

Pygame is required to use module `simplegui_lib` (and its submodules) and module `simpleguics2pygame` of SimpleGUICS2Pygame (except for the Timer class).

Warning: Normally Pygame is installed automatically when you install SimpleGUICS2Pygame. But if it is failed then install it like this first.

```
$ python -m pip install pygame --user --upgrade
```

If you have some problem, see installation documentation of pygame.

On Window$ you can also directly install a binary from the Unofficial Windows Binaries for Python Extension Packages: Pygame binary.

2.4 Package audioread required

audioread is required to play MP3 sounds (other sounds are played by Pygame).

Warning: If audioread is correctly installed on your system but playing MP3 fails, it is probably because some external dependency is missing. Try to install FFmpeg (executable `ffmpeg` must be accessible in your path) or other library used by audioread.

2.5 Package matplotlib required

matplotlib is required to use module `simpleplot` of SimpleGUICS2Pygame.

If you have some problem, see installation documentation of matplotlib.

On Window$ you can also directly install matplotlib binary.

2.6 Test installation

You can run the little script `SimpleGUICS2Pygame_check.py` to check if all required modules are installed.

Examples of result with good installation: result in Python 2 and result in Python 3.

You can also test your Pygame installation alone with the other little script `pygame_check.py`.

CHAPTER 3

Examples of CodeSkulptor and SimpleGUICS2Pygame use

You can see examples in SimpleGUICS2Pygame/example/ subdirectory from the sources archives.

Or online: Python programs running in CodeSkulptor .

On some environments CodeSkulptor maybe runs too slow with Firefox. CodeSkulptor3 is better, but maybe too slow too. In these cases, use Chrome.

Two simple online examples:

- Frame_example.py: very simple canvas example
- presentation.py: little draw images and texts

CHAPTER 4

Message to developers

This is a **free software**, so you can download it, **modify it** and **submit your modifications**. You can also **redistribute** your own version (keeping the GPL license).

Complete **sources** on Bitbucket: https://bitbucket.org/OPiMedia/simpleguics2pygame

See developers'page.

CHAPTER 5

Author: Olivier Pirson — OPi 

Website: http://www.opimedia.be/

Bitbucket: https://bitbucket.org/OPiMedia/

- olivier.pirson.opi@gmail.com
- Mastodon: https://mamot.fr/@OPiMedia — Twitter: https://twitter.com/OPirson
- diaspora* (Framasphere*): https://framasphere.org/u/opimedia
- LinkedIn: https://www.linkedin.com/in/olivierpirson/ — CV: http://www.opimedia.be/CV/English.html
- other profiles: http://www.opimedia.be/about/

CHAPTER 6

Support me

This program is a **free software** (GPL license). It is **completely free** (like "free speech" *and* like "free beer"). However you can **support me** financially by donating.

Click to this link 

Thank you!

CHAPTER 7

Note that

- SimpleGUI of CodeSkulptor (Scott Rixner) is a specific module of CodeSkulptor, written in JavaScript.

  CodeSkulptor is a Python implementation running **in a browser**. It implements a subset of Python **2**. It is the environment used in the course An Introduction to Interactive Programming in Python (Rice University, Coursera).

- SimpleGUI of CodeSkulptor3 (Scott Rixner) is the same in the new version CodeSkulptor3 that implements a subset of Python **3**.

- **SimpleGUICS2Pygame** (Olivier Pirson) is **this package**. It is fully compatible with Python **2 and 3**.

  It contains `codeskulptor`, `numeric`, `simpleguics2pygame` and `simpleplot` modules that reimplement `codeskulptor`, `numeric`, `simplegui` and `simpleplot` modules of CodeSkulptor.

  `simplemap` is *not* implemented.

  > **Warning:** SimpleGUICS2Pygame was **designed to mimic behavior of CodeSkulptor**. So load_image() and load_sound() methods can load medias only from URL, not local files. However SimpleGUICS2Pygame can save these medias to a specific local directory. See the Download medias tips.
  >
  > You can also use *specific* _load_local_image() and _load_local_sound() methods to load local files. But be careful, each specific method doesn't exist in CodeSkulptor.
  >
  > There exist some **little differences between SimpleGUICS2Pygame and SimpleGUI** of CodeSkulptor. See Compatibility notes.

- SimpleGUITk (David Holm) is *another implementation* of SimpleGUI of CodeSkulptor, using Tkinter and some others packages. It is really less complete and not updated. However it works for some programs.

- simplegui2pygamemodule (Jimmy Kumar Ahalpara) seems be *another implementation* of SimpleGUI of CodeSkulptor, but I have not tested it.

- simplequi (Arthur Gordon-Wright) is *another implementation* of SimpleGUI of CodeSkulptor, using Qt/PySide2. It is a partial implementation that I have not tested.

Warning:

- simplegui (Florian Berger) is a Python package which has the same name as SimpleGUI of CodeSkulptor, but it is *totally something else*.
- PySimpleGUI is also a Python package that is *totally something else*.

CHAPTER 8

Table of contents

8.1 Package SimpleGUICS2Pygame

SimpleGUICS2Pygame package.

It is primarily a standard Python (**2 and 3**) module reimplementing the SimpleGUI particular module of CodeSkulptor and CodeSkulptor3 (a Python browser environment). This is in fact a package also with other modules adapted from CodeSkulptor.

Require Pygame (except for the Timer class) (and must be installed separately).

Module simpleplot require matplotlib .

Online HTML documentation on Read The Docs.

Sources and installers on Bitbucket: https://bitbucket.org/OPiMedia/simpleguics2pygame
and on PyPI: https://pypi.org/project/SimpleGUICS2Pygame/ .

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2016, 2018, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version November 29, 2020

- v.2.1.0 — November 29, 2020
- v.2.0.3 — October 2, 2020
- v.2.0.2 — May 23, 2020
- v.2.0.1 — May 21, 2020
- v.2.0.0 — April 18, 2020
- v.01.09.00 — January 1st, 2015

- v.01.08.01 — October 9, 2014
- v.01.08.00 — October 4, 2014
- v.01.07.00 — September 2, 2014
- v.01.06.03 — July 24, 2014
- v.01.06.02 — July 18, 2014
- v.01.06.01 — July 17, 2014
- v.01.06.00 — June 16, 2014
- v.01.05.00 — May 25, 2014
- v.01.04.00 — December 16, 2013
- v.01.03.00 — December 13, 2013
- v.01.02.00 — November 8, 2013
- v.01.01.00 — November 1st, 2013
- v.01.00.02 — October 31, 2013
- v.01.00.01 — October 9, 2013
- v.01.00.00 — July 13, 2013
- v.00.92.00 — June 27, 2013
- v.00.91.00 — June 23, 2013
- v.00.90.10 — June 19, 2013
- v.00.90.00 — June 13, 2013
- Started on May 21, 2013

Complete changelog

`SimpleGUICS2Pygame.__init__._VERSION = '2.1.0'`
Version of SimpleGUICS2Pygame package.

`SimpleGUICS2Pygame.__init__._WEBSITE = 'https://bitbucket.org/OPiMedia/simpleguics2pygame/`
Website of the project.

`SimpleGUICS2Pygame.__init__._WEBSITE_DOC = 'https://simpleguics2pygame.readthedocs.io/'`
Website of the documentation.

8.2 All modules of this package

8.2.1 codeskulptor — replace the codeskulptor module of CodeSkulptor

codeskulptor module.

Replace the codeskulptor module of CodeSkulptor.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2014, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 19, 2020

SimpleGUICS2Pygame.codeskulptor.**file2url**(*filename*)

Return a completed CodeSkulptor URL ressource from a short *filename*.

Example given in the CodeSkulptor file2url documentation: *file2url('assets-Quick_fox.txt')* returns *'http://codeskulptor-assets.commondatastorage.googleapis.com/assets-Quick_fox.txt'*

Example given in the CodeSkulptor3 urllib2-urlopen documentation (there is an error in *file2url* documentation): *file2url('assets_sample_text.txt')* returns *'//codeskulptor-assets.commondatastorage.googleapis.com/assets_sample_text.txt'*

Parameters **filename** – str

Raise ValueError if filename is in a incorrect format (the good format is '^[a-zA-Z][a-zA-Z0-9]*[_-]')

Returns str

SimpleGUICS2Pygame.codeskulptor.**randomize_iteration**(*randomize=True*)

Fake implementation. In CodeSkulptor this function modify the default behaviour of iterations on `dict` and `set`. In SimpleGUICS2Pygame this function does nothing.

See CodeSkulptor3 randomize_iteration documentation.

(Available in CodeSkulptor but *not in CodeSkulptor documentation*!)

Randomize bool

SimpleGUICS2Pygame.codeskulptor.**set_timeout**(*seconds*)

Do nothing.

In CodeSkulptor, this function change the timeout imposed on all programs (by default 5 seconds). See CodeSkulptor set_timeout documentation.

(Available in CodeSkulptor and CodeSkulptor3 but *not in CodeSkulptor3 documentation*!)

Parameters **seconds** – int >= 0

[source]

8.2.2 codeskulptor_lib — some miscellaneous functions

(Version saved in CodeSkulptor https://py3.codeskulptor.org/#user305_SXBsmszNiUxIeoV.py .)

codeskulptor_lib module.

Some miscellaneous functions to help in CodeSkulptor.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2014, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 19, 2020

SimpleGUICS2Pygame.codeskulptor_lib.**__CODESKULPTOR_IS = None**

Used to memoization by codeskulptor_is().

SimpleGUICS2Pygame.codeskulptor_lib.**__CODESKULPTOR_VERSION = None**

Used to memoization by codeskulptor_version().

SimpleGUICS2Pygame.codeskulptor_lib.**assert_position**(*position*, *non_negative=False*, *non_zero=False*)

Assertions to check valid *position*.

If non_negative then each *int* or *float* must be >= 0.

If non_zero then each *int* or *float* must be != 0.

Parameters

- **position** – (int or float, int or float) or [int or float, int or float]
- **non_negative** – bool

SimpleGUICS2Pygame.codeskulptor_lib.**codeskulptor_is**()
If run in CodeSkulptor environment then return *True*, else return *False*.

Returns bool

SimpleGUICS2Pygame.codeskulptor_lib.**codeskulptor_version**()
If run in CodeSkulptor environment then return 2 if CodeSkulptor or 3 if CodeSkulptor3 else return *False*.

Returns False, 2 or 3

SimpleGUICS2Pygame.codeskulptor_lib.**hex2**(*n*, *uppercase=True*)
Return 2 characters corresponding to the hexadecimal representation of *n*.

Parameters

- **n** – 0 <= int < 256
- **uppercase** – bool

Returns str (length == 2)

SimpleGUICS2Pygame.codeskulptor_lib.**hex_fig**(*n*, *uppercase=True*)
Return the hexadecimal figure of *n*.

Parameters

- **n** – 0 <= int < 16
- **uppercase** – bool

Returns str (one character from 0123456789ABCDEF or 0123456789abcdef)

SimpleGUICS2Pygame.codeskulptor_lib.**hsl**(*hue*, *saturation*, *lightness*)
Return the string HTML representation of the color in 'hsl(hue, lightness, saturation)' format.

Parameters

- **hue** – float or int
- **saturation** – 0 <= float or int <= 100
- **lightness** – 0 <= float or int <= 100

Returns str

SimpleGUICS2Pygame.codeskulptor_lib.**hsla**(*hue*, *saturation*, *lightness*, *alpha=1*)
Return the string HTML representation of the color in 'hsla(hue, lightness, saturation, alpha)' format.

Parameters

- **hue** – float or int
- **saturation** – 0 <= float or int <= 100
- **lightness** – 0 <= float or int <= 100
- **alpha** – 0 <= float or int <= 1

Returns str

SimpleGUICS2Pygame.codeskulptor_lib.**rgb**(*red*, *green*, *blue*)
Return the string HTML representation of the color in 'rgb(red, blue, green)' format.

Parameters

- **red** – 0 <= int <= 255
- **green** – 0 <= int <= 255
- **blue** – 0 <= int <= 255

Returns str

SimpleGUICS2Pygame.codeskulptor_lib.**rgba**(*red*, *green*, *blue*, *alpha=1*)
Return the string HTML representation of the color in 'rgba(red, blue, green, alpha)' format.

Parameters

- **red** – 0 <= int <= 255
- **green** – 0 <= int <= 255
- **blue** – 0 <= int <= 255
- **alpha** – 0 <= float or int <= 1

Returns str

[source]

8.2.3 numeric — replace the numeric module of CodeSkulptor

numeric module.

Replace the numeric module of CodeSkulptor.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2014, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 20, 2020

class SimpleGUICS2Pygame.numeric.**Matrix**(*data*, *_copy=True*)
Matrix (m x n).

See http://en.wikipedia.org/wiki/Matrix_%28mathematics%29 .

__add__(*other*)
To a matrix (m x n) return the matrix plus other.

Parameters **other** – Matrix (m x n)

Returns Matrix (m x n)

__getitem__(*i_j*)
Return the value of the (m x n) matrix at row i and column j.

Parameters **i_j** – (0 <= int < m, 0 <= int < n) or [0 <= int < m, 0 <= int < n]

Returns float

__init__(*data*, *_copy=True*)
Create a matrix with the 2-dimensional *data*.

If not *_copy* then *data* is directly used without copy. In this case, *data* must be a correct list of list of float. **(Option not available in SimpleGUI of CodeSkulptor.)**

Parameters

- **data** – (not empty tuple or list) of (same size tuple or list) of (int or float)
- **_copy** – bool

__mul__(*other*)
To a matrix (m x k) return the matrix multiply by other.

Parameters **other** – Matrix (k x n)

Returns Matrix (m x n)

__setitem__(*i_j*, *value*)
Change the value of the element at row i and column j, to the (m x n) matrix.

Parameters

- **i_j** – (0 <= int < m, 0 <= int < n) or [0 <= int < m, 0 <= int < n]
- **value** – int or float

__str__()
Return the string representation of the matrix.

Returns string

__sub__(*other*)
To a matrix (m x n) return the matrix minus other.

Parameters **other** – Matrix (m x n)

Returns Matrix (m x n)

__weakref__
list of weak references to the object (if defined)

_is_identity(*epsilon=2.220446049250313e-16*)
If the matrix is an identity matrix then return True, else return False.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **epsilon** – 0 <= (float or int) < 1

Returns bool

_is_zero(*epsilon=2.220446049250313e-16*)
If the matrix is a zeros matrix then return True, else return False.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **epsilon** – 0 <= (float or int) < 1

Returns bool

_nb_columns()
Return n for a (m x n) matrix.

(Not available in SimpleGUI of CodeSkulptor.)

Returns int >= 1

_nb_lines()
Return m to a (m x n) matrix.

(Not available in SimpleGUI of CodeSkulptor.)

Returns int >= 1

abs()
To a matrix (m x n) return the matrix with each element is the absolute value.

Returns Matrix (m x n)

copy()
Return a copy of the matrix (m x n).

Returns Matrix (m x n)

getcol(*j*)
Return the (1 x m) matrix that is a copy of column j of the (m x n) matrix.

Parameters **j** – 0 <= int < n

Returns Matrix (1 x m)

getrow(*i*)
Return the (1 x n) matrix that is a copy of row i of the (m x n) matrix.

Parameters **i** – 0 <= int < m

Returns Matrix (1 x n)

inverse(*_epsilon=2.220446049250313e-16*)
If the square matrix (n x n) is inversible then return the inverse, else raise an ValueError exception.

Algorithm used: Gaussian elimination. See http://en.wikipedia.org/wiki/Gaussian_elimination .

Parameters **_epsilon** – 0 <= (float or int) < 1 **(Option not available in SimpleGUI of CodeSkulptor.)**

Returns Matrix (n x n)

Raise ValueError if the matrix is not inversible

scale(*factor*)
To a matrix (m x n) return the matrix with each element multiply by factor.

(Method available in CodeSkulptor 2 but not in CodeSkulptor!)

Parameters **factor** – int or float

Returns Matrix (m x n)

shape()
Return (m, n) to a matrix (m x n).

Returns (int >= 1, int >= 1)

summation()
Return the sum of all the elements of the matrix.

Returns float

transpose()
Return the transposition of the matrix (m x n).

Returns Matrix (n x m)

SimpleGUICS2Pygame.numeric.**_EPSILON = 2.220446049250313e-16**
The default epsilon value.

SimpleGUICS2Pygame.numeric.**_zero**(*m*, *n*)
Return a (*m* x *n*) zeros matrix.

Parameters

- **m** – int >= 1
- **n** – int >= 1

Returns Matrix (*m* x *n*)

SimpleGUICS2Pygame.numeric.**identity**(*size*)
Return a (*size* x *size*) identity matrix.

Parameters **size** – int >= 1

Returns Matrix (*size* x *size*)

[source]

8.2.4 simplegui_lib — simply import the following modules

simplegui_lib_draw — draw functions

(Version saved in CodeSkulptor https://py3.codeskulptor.org/#user305_SaT1YKoOikl4ax9.py .)

simplegui_lib_draw module.

Draw functions to help in SimpleGUI of CodeSkulptor.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013, 2015, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 19, 2020

SimpleGUICS2Pygame.simplegui_lib_draw.**draw_rect**(*canvas*, *pos*, *size*, *line_width*, *line_color*, *fill_color=None*)
Draw a rectangle.

Parameters

- **canvas** – simplegui.Canvas
- **pos** – (int or float, int or float) or [int or float, int or float]
- **size** – (int or float, int or float) or [int or float, int or float]
- **line_width** – int >= 0
- **line_color** – str
- **fill_color** – str

SimpleGUICS2Pygame.simplegui_lib_draw.**draw_text_multi**(*canvas*, *text*, *point*, *font_size*, *font_color*, *font_face='serif'*, *_font_size_coef=0.75*)
Draw the *text* (possibly with several lines) at the position *point*.

If *text* is a str, then split it on each end of line.

If *text* is a tuple or a list of str, then print each str on a separated line.

See simplegui.draw_text() .

Parameters

- **canvas** – simplegui.Canvas
- **text** – str or (tuple of str) or (list of str)
- **point** – (int or float, int or float) or [int or float, int or float]
- **font_size** – (int or float) >= 0
- **font_color** – str
- **font_face** – str == 'monospace', 'sans-serif', 'serif'
- **_font_size_coef** – int or float

Raise ValueError if text contains unprintable whitespace character

SimpleGUICS2Pygame.simplegui_lib_draw.**draw_text_side**(*frame*, *canvas*, *text*, *point*, *font_size*, *font_color*, *font_face='serif'*, *font_size_coef=0.75*, *rectangle_color=None*, *rectangle_fill_color=None*, *side_x=-1*, *side_y=1*)

Draw the *text* string at the position *point*.

See simplegui.draw_text() .

If *rectangle_color* != *None* then draw a rectangle around the text.

If *rectangle_fill_color* != *None* then draw a filled rectangle under the text.

If *side_x*
< 0 then *point[0]* is the left of the text,
== 0 then *point[0]* is the center of the text,
> 0 then *point[0]* is the right of the text.

If *side_y*
< 0 then *point[1]* is the top of the text,
== 0 then *point[1]* is the center of the text,
> 0 then *point[1]* is the bottom of the text.

Parameters

- **frame** – simplegui.Frame
- **canvas** – simplegui.Canvas
- **text** – str
- **point** – (int or float, int or float) or [int or float, int or float]
- **font_size** – (int or float) >= 0
- **font_color** – str
- **font_face** – str == 'monospace', 'sans-serif', 'serif'
- **font_size_coef** – int or float

- **rectangle_color** – None or str
- **rectangle_fill_color** – None or str
- **side_x** – int or float
- **side_y** – int or float

[source]

simplegui_lib_fps — class to calculate and display Frames Per Second

(Version saved in CodeSkulptor https://py3.codeskulptor.org/#user305_tXfH4AcbNLtjfHy.py .)

Examples of use in :

- *test/test_image.py*: https://py3.codeskulptor.org/#user305_pFn2dOrRPLhl8Z4.py
- *example/Spaceship_prototype.py*: https://py3.codeskulptor.org/#user305_oBWI7SgNVos3Lgx.py
- *example/RiceRocks_Asteroids.py*: https://py3.codeskulptor.org/#user305_XNvcqTxIBngtHPu.py

simplegui_lib_fps module.

A class to calculate and display FPS (Frames Per Second) in SimpleGUI of CodeSkulptor.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2014, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 19, 2020

class SimpleGUICS2Pygame.simplegui_lib_fps.**FPS**(*x=10*, *y=10*, *font_color='Red'*, *font_size=40*)

Calculate and display FPS (Frames Per Second).

How to use:

- Create an instance of FPS: `fps = FPS()`
- Start: `fps.start()`
- And put the `draw_fct()` in the end of your canvas' draw handler: `fps.draw_fct(canvas)`

__init__(*x=10*, *y=10*, *font_color='Red'*, *font_size=40*)

Set an instance to calculate FPS and drawing on position (x, y).

Parameters

- **x** – int or float
- **y** – int or float
- **font_color** – str
- **font_size** – int > 0

__weakref__

list of weak references to the object (if defined)

draw_fct(*canvas*)

Update the number of frames drawn and draw the FPS.

This method **must be** called from the canvas' draw handler (the function passed as a parameter to *simplegui.Frame.set_draw_handler()*).

Parameters **canvas** – simplegui.Canvas

is_started()
If FPS is active then return True, else return False.

start()
Start calculation and drawing.

See *draw_fct()*.

stop()
Stop calculation and drawing.

[source]

simplegui_lib_keys — class to manage keyboard handling

(Version saved in CodeSkulptor https://py3.codeskulptor.org/#user305_EtIUDiM87dN1mD2.py .)

Examples of use in :

- *example/keys.py*: https://py3.codeskulptor.org/#user305_y6XS9Bq5JFD4eOU.py

simplegui_lib_keys module.

A class to help manage keyboard handling in SimpleGUI of CodeSkulptor.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2014, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 19, 2020

class SimpleGUICS2Pygame.simplegui_lib_keys.**Keys**(*frame*, *keys=None*)
Keys handler.

Set and catch keys handlers of SimpleGUICS2Pygame (and CodeSkulptor) to help.

General note: Some keyboards can't handle more than two or three keys pressed simultaneously. See Keyboard Ghosting Explained! and Keyboard Ghosting Demonstration.

__init__(*frame*, *keys=None*)
If keys is None then set an empty keys handler, else set a keys handler with key up and key down functions of keys.

active_handlers(), *active_keydown_handler()* or *active_keyup_handler()* must be called to activate.

Parameters

- **frame** – simplegui.Frame
- **keys** – None or Keys

__weakref__
list of weak references to the object (if defined)

active_handlers()
Active key down and key up handlers.

active_keydown_handler()
Active the key down handler.

active_keyup_handler()
Active the key up handler.

is_pressed(*key_code*)
If the key is pressed then return True, else return False.

Parameters **key_code** – int >= 0

Returns bool

is_pressed_key_map(*key_str*)
If the key is pressed then return True, else return False.

Parameters **key_str** – str in *simplegui.KEY_MAP*

Returns bool

pressed_keys()
Return a sorted list with code of all pressed keys.

Returns list of (int >= 0)

set_keydown_fct(*key_code*, *fct=None*)
If fct is None then erase the function key down handler to the specified key, else set the function key down handler to the specified key.

Parameters

- **key_code** – int >= 0
- **fct** – (int) -> *

set_keydown_fct_key_map(*key_str*, *fct=None*)
If fct is None then erase the function key down handler to the specified key, else set the function key down handler to the specified key.

Parameters

- **key_str** – str in *simplegui.KEY_MAP*
- **fct** – (int) -> *

set_keyup_fct(*key_code*, *fct=None*)
If fct is None then erase the function key up handler to the specified key, else set the function key up handler to the specified key.

Parameters

- **key_code** – int >= 0
- **fct** – (int) -> *

set_keyup_fct_key_map(*key_str*, *fct=None*)
If fct is None then erase the function key up handler to the specified key, else set the function key up handler to the specified key.

Parameters

- **key_str** – str in *simplegui.KEY_MAP*
- **key_code** – int >= 0
- **fct** – (int) -> *

[source]

simplegui_lib_loader — class to load images and sounds

(Version saved in CodeSkulptor https://py3.codeskulptor.org/#user305_SZPJfNxJlVTjbAy.py .)

Examples of use in :

- *example/loader.py*: https://py3.codeskulptor.org/#user305_vyR7lzNAMOV0BbV.py
- *test/test_image.py*: https://py3.codeskulptor.org/#user305_pFn2dOrRPLhl8Z4.py
- *example/Spaceship_prototype.py*: https://py3.codeskulptor.org/#user305_oBWI7SgNVos3Lgx.py
- *example/RiceRocks_Asteroids.py*: https://py3.codeskulptor.org/#user305_XNvcqTxIBngtHPu.py

simplegui_lib_loader module.

A class to help load images and sounds in SimpleGUI of CodeSkulptor.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2015, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 19, 2020

class SimpleGUICS2Pygame.simplegui_lib_loader.**Loader**(*frame*, *progression_bar_width*, *after_function*, *max_waiting=5000*)

Help to load images and sounds from Internet and wait finished.

With SimpleGUICS2Pygame, *SimpleGUICS2Pygame.load_image()* and *SimpleGUICS2Pygame.load_sound()* wait automatically until loading is completed.

But in CodeSkulptor, the browser load images and sounds asynchronously. (With SimpleGUI it is **impossible to verify that the sounds are loaded**. So *Loader* begin load sounds, and next begin load images. It wait each image is loaded, and considers that all downloads are completed.)

__SIMPLEGUICS2PYGAME = None
True if SimpleGUICS2Pygame are used, else *False*.

__init__(*frame*, *progression_bar_width*, *after_function*, *max_waiting=5000*)
Set an empty loader.

Parameters

- **frame** – simplegui.Frame
- **progression_bar_width** – (int or float) >= 0
- **after_function** – function () -> *
- **max_waiting** – (int or float) >= 0

__weakref__
list of weak references to the object (if defined)

_draw_loading(*canvas*)
Draw waiting message on the canvas when images and sounds loading.

Parameters **canvas** – simplegui.Canvas

_interval = 100
Interval in ms betweed two check.

add_image(*url*, *name=None*)
Add an image from *url* and give it a name.

Execute 'Loader.load()' before use images.

If *name* == *None* then "filename" of url is used.

Example: If *url* == *'http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/asteroid_blue.png'* and *name* == *None* then *'asteroid_blue.png'* is used.

Parameters

- **url** – str
- **name** – None or str

add_sound(*url*, *name=None*)
Add a sound from *url* and give it a *name*.

Execute 'Loader.load()' before use sounds.

If *name* == *None* then "filename" of *url* is used.

Example: If *url* == *'http://commondatastorage.googleapis.com/codeskulptor-assets/Epoq-Lepidoptera.ogg'* and *name* == *None* then *'Epoq-Lepidoptera.ogg'* is used.

Parameters

- **url** – str
- **name** – None or str

cache_clear()

- In standard Python with SimpleGUICS2Pygame: Empty the cache of Pygame surfaces used by each image of this Loader. See Image._pygamesurfaces_cached_clear .
- In SimpleGUI of CodeSkulptor: do nothing.

get_image(*name*)
If an image named *name* exist then return it, else return *None*

Parameters **name** – str

Raise Exception if Loader.load() was not executed since the addition of this image.

Returns None or simplegui.Image

get_nb_images()
Return the number of images (loaded or not).

Returns int >= 0

get_nb_images_loaded()
Return the number of loaded images.

It is the number of begin loading by *Loader.load()* **and** fully completed.

Returns int >= 0

get_nb_sounds()
Return the number of sounds (loaded or not).

Returns int >= 0

get_nb_sounds_loaded()
Return the number of loaded sounds.

It is the number of begin loading by *Loader.load()*, **but not necessarily completed**. Because with SimpleGUI of CodeSkulptor it is **impossible to verify that the sounds are loaded**.

Returns int >= 0

get_sound(*name*)
If a sound named *name* exist then return it, else return *None*

Parameters **name** – str

Raise Exception if load() was not executed since the addition of this sound.

Returns None or simplegui.Sound

load()
Start loading of all images and sounds added since last *Loader.load()* execution.

- In standard Python with SimpleGUICS2Pygame: draw a progression bar on canvas and wait until the loading is finished.
- In SimpleGUI of CodeSkulptor: *don't* wait.

pause_sounds()
Pause all sounds.

print_stats_cache()

- In standard Python with SimpleGUICS2Pygame: Print to stderr some statistics of cached Pygame surfaces used by each image of this Loader. See Image._print_stats_cache .
- In SimpleGUI of CodeSkulptor: do nothing.

wait_loaded()
Draw a progression bar on canvas and wait until all images and sounds are fully loaded. Then execute *self._after_function*.

After *self._max_waiting* milliseconds, abort and execute *self._after_function*.

See details in *get_nb_sounds_loaded()* documentation.

[source]

(Version saved in CodeSkulptor https://py3.codeskulptor.org/#user305_SZNWcbqQHXN4pow.py .)

simplegui_lib module.

Some functions and classes to help in SimpleGUI of CodeSkulptor, from *simplegui_lib_draw*, *simplegui_lib_fps*, *simplegui_lib_keys* and *simplegui_lib_loader*.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2015, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 21, 2020

8.2.5 simpleguics2pygame — the main module, replace the simplegui module of CodeSkulptor

simpleguics2pygame — canvas

simpleguics2pygame module: simpleguics2pygame/canvas.

Class Canvas.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2015-2016, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version November 29, 2020

class SimpleGUICS2Pygame.simpleguics2pygame.canvas.**Canvas**(*frame*, *canvas_width*, *canvas_height*)

Canvas similar to SimpleGUI *Canvas* of CodeSkulptor.

__init__(*frame*, *canvas_width*, *canvas_height*)

Set the canvas.

Don't use directly, a canvas is created by *Frame()* and reachable by handler defined by *Frame.set_draw_handler()*.

Parameters

- **frame** – Frame (or None)
- **canvas_width** – int >= 0
- **canvas_height** – int >= 0

__repr__()

Return *'<Canvas object>'*.

Returns str

__weakref__

list of weak references to the object (if defined)

_background_pygame_color

Default *pygame.Color* of the background of the canvas.

_bg_pygame_surface_image = None

pygame.surface.Surface default background image replaces *_background_pygame_color*.

_draw()

If *self._draw_handler* != *None* then call it and update display of the canvas.

(Not available in SimpleGUI of CodeSkulptor.)

_save(*filename*)

Save the canvas in *filename*.

Supported formats are supported formats by Pygame to save: TGA, PNG, JPEG or BMP (see https://www.pygame.org/docs/ref/image.html#pygame.image.save).

If *filename* extension is not recognized then TGA format is used.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **filename** – str

draw_arc(*center_point*, *radius*, *start_angle*, *end_angle*, *line_width*, *line_color*)

Draw an arc of circle, from *start_angle* to *end_angle*. Angles given in radians are clockwise and start from 0 at the 3 o'clock position.

(Available in CodeSkulptor3 but *not in CodeSkulptor 2*!)

Parameters

- **center_point** – (int or float, int or float) or [int or float, int or float]
- **radius** – (int or float) > 0

- **start_angle** – int or float
- **end_angle** – int or float
- **line_width** – (int or float) > 0
- **line_color** – str

draw_circle(*center_point*, *radius*, *line_width*, *line_color*, *fill_color=None*)
Draw a circle.

If *fill_color* != *None* then fill with this color.

Parameters

- **center_point** – (int or float, int or float) or [int or float, int or float]
- **radius** – (int or float) > 0
- **line_width** – (int or float) > 0
- **line_color** – str
- **fill_color** – None or str

draw_image(*image*, *center_source*, *width_height_source*, *center_dest*, *width_height_dest*, *rotation=0*)
Draw *image* on the canvas.

Specify center position and size of the source (*image*) and center position and size of the destination (the canvas).

Size of the source allow get a piece of *image*. If *width_height_source* is bigger than *image* then draw nothing.

Size of the destination allow rescale the drawed image.

rotation specify a clockwise rotation in radians.

Each new Pygame surface used is added to *image._pygamesurfaces_cached*. See Image._pygamesurfaces_cached_clear() .

If number of surfaces in this caches is greater than *image._pygamesurfaces_cache_max_size* then remove the oldest surface.

Parameters

- **image** – Image
- **center_source** – (int or float, int or float) or [int or float, int or float]
- **width_height_source** – ((int or float) >= 0, (int or float) >= 0) or [(int or float) >= 0, (int or float) >= 0]
- **center_dest** – (int or float, int or float) or [int or float, int or float]
- **width_height_dest** – ((int or float) >= 0, (int or float) >= 0) or [(int or float) >= 0, (int or float) >= 0]
- **rotation** – int or float

draw_line(*point1*, *point2*, *line_width*, *line_color*)
Draw a line segment from point1 to point2.

Parameters

- **point1** – (int or float, int or float) or [int or float, int or float]
- **point2** – (int or float, int or float) or [int or float, int or float]

- **line_width** – (int or float) > 0
- **line_color** – str

draw_point(*position*, *color*)
Draw a point.

Parameters

- **position** – (int or float, int or float) or [int or float, int or float]
- **color** – str

draw_polygon(*point_list*, *line_width*, *line_color*, *fill_color=None*)
Draw a polygon from a list of points. A segment is automatically drawed between the last point and the first point.

If *fill color* is not None then fill with this color.

If *line_width* > 1, ends are poorly made!

Parameters

- **point_list** – not empty (tuple or list) of ((int or float, int or float) or [int or float, int or float])
- **line_width** – (int or float) > 0
- **line_color** – str
- **fill_color** – None or str

draw_polyline(*point_list*, *line_width*, *line_color*)
Draw line segments between a list of points.

If *line_width* > 1, ends are poorly made!

Parameters

- **point_list** – not empty (tuple or list) of ((int or float, int or float) or [int or float, int or float])
- **line_width** – (int or float) > 0
- **line_color** – str

draw_text(*text*, *point*, *font_size*, *font_color*, *font_face='serif'*, *_font_size_coef=0.75*)
Draw the *text* string at the position *point*.

(*point[0]* is the left of the text, *point[1]* is the bottom of the text.)

If correponding font in Pygame is not founded, then use the default *pygame.font.Font*.

_font_size_coef is used to adjust the vertical positioning. **(This paramater is not available in SimpleGUI of CodeSkulptor.)**

Warning This method can't draw multiline text.

To draw multiline text, see simplegui_lib_draw.draw_text_multi() .

Parameters

- **text** – str
- **point** – (int or float, int or float) or [int or float, int or float]
- **font_size** – (int or float) >= 0
- **font_color** – str

- **font_face** – str == 'monospace', 'sans-serif', 'serif'
- **_font_size_coef** – int or float

Raise ValueError if text contains unprintable whitespace character

(Alpha color channel don't work!!!)

SimpleGUICS2Pygame.simpleguics2pygame.canvas.**create_invisible_canvas**(*width*, *height*)

NOT IMPLEMENTED! (Return a "weak" *Canvas*.)

(Available in SimpleGUI of CodeSkulptor but *not in CodeSkulptor documentation*!)

Parameters

- **width** – int >= 0
- **height** – int >= 0

Returns Canvas

SimpleGUICS2Pygame.simpleguics2pygame.canvas.**__all__ = ('Canvas', 'create_invisible_canvas**

Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

[source]

simpleguics2pygame — control

simpleguics2pygame module: simpleguics2pygame/control.

Classes Control and TextAreaControl.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2015-2016, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version November 29, 2020

class SimpleGUICS2Pygame.simpleguics2pygame.control.**Control**(*frame*, *text*, *button_handler=None*, *width=None*)

Control similar to SimpleGUI *Control* (button and label) of CodeSkulptor.

__init__(*frame*, *text*, *button_handler=None*, *width=None*)

Set a button (if button_handler is not None) or a label (if button_handler is None) in the control panel.

Don't use directly, use *Frame.add_button()* or *Frame.add_label()*.

Parameters

- **frame** – Frame
- **text** – str
- **button_handler** – None or (function () -> *)
- **width** – None or int

__repr__()
Return *'<Control object>'*.

Returns str

__weakref__
list of weak references to the object (if defined)

_button_background_pygame_color
pygame.Color of the background in the button.

_button_padding_x = 5
Horizontal padding in the button.

_button_padding_y = 3
Vertical padding in the button.

_button_pygame_font
pygame.font.Font of text in the button.

_button_selected_background_pygame_color
pygame.Color of the background in the button when it has pressed.

_button_text_pygame_color
pygame.Color of text in the button.

_draw()
Draw the control object in the control panel.

(Not available in SimpleGUI of CodeSkulptor.)

_draw_button()
Draw the the control object as a button.

(Not available in SimpleGUI of CodeSkulptor.)

_draw_label()
Draw the the control object as a label.

(Not available in SimpleGUI of CodeSkulptor.)

_label_pygame_font
pygame.font.Font of the label.

_label_text_pygame_color
pygame.Color of the label.

_mouse_left_button(*pressed*)
Deal a click of left mouse button on the zone of this *Control*.

If *pressed* then select this Control, else unselect and run the button handler (if exist).

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **pressed** – bool

_pos_in(*x*, *y*)
If position (*x*, *y*) is on the zone of this *aControl* then return *True*, else return *False*.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters

- **x** – int or float
- **y** – int or float

Returns bool

get_text()
Return the text of the button or the label.

Returns str

set_text(*text*)
Change the text of the button or the label.

Parameters **text** – str

class SimpleGUICS2Pygame.simpleguics2pygame.control.**TextAreaControl**(*frame*, *label_text*, *input_handler*, *input_width*)
TextAreaControl similar to SimpleGUI *TextAreaControl* (input) of CodeSkulptor.

__init__(*frame*, *label_text*, *input_handler*, *input_width*)
Set a input box in the control panel.

Don't use directly, use *Frame.add_input()*.

Parameters

- **frame** – Frame
- **label_text** – str
- **input_handler** – function (str) -> *
- **input_width** – int or float

__repr__()
Return *'<TextAreaControl object>'*.

Returns str

__weakref__
list of weak references to the object (if defined)

_draw()
Draw the input box and his label.

(Not available in SimpleGUI of CodeSkulptor.)

_input_background_pygame_color
pygame.Color of the background in the input box.

_input_mark_pygame_color
pygame.Color of the end mark of text in the input box.

_input_padding_x = 5
Horizontal padding in the input box.

_input_padding_y = 3
Vertical padding in the input box.

_input_pygame_color
pygame.Color of the text in the input box.

_input_pygame_font
pygame.font.Font of the text in the input box.

_input_selected_background_pygame_color
pygame.Color of the background in the input box when it has focus.

_key (*pygame_event*)
Deal key pressed when this *TextAreaControl* have focus.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **pygame_event** – pygame.Event KEYDOWN or KEYUP

_label_pygame_font
pygame.font.Font of the label of the input box.

_label_text_pygame_color
pygame.Color of the label of the input box.

_mouse_left_button (*pressed*)
Deal a click of left mouse button on the zone of this *TextAreaControl*.

If *pressed* then give it the focus.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **pressed** – bool

_pos_in (*x*, *y*)
If position (*x*, *y*) is on the zone of this *TextAreaControl* then return *True*, else return *False*.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters

- **x** – int or float
- **y** – int or float

Returns bool

get_text ()
Return the text of the input box.

Returns str (or unicode in Python 2)

set_text (*input_text*)
Change the text in the input box.

Parameters **input_text** – str

SimpleGUICS2Pygame.simpleguics2pygame.control.**__all__ = ('Control', 'TextAreaControl')**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

[source]

simpleguics2pygame — frame

simpleguics2pygame module: simpleguics2pygame/frame.

Class Frame.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2015-2016, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version November 29, 2020

class SimpleGUICS2Pygame.simpleguics2pygame.frame.**Frame**(*title*, *canvas_width*, *canvas_height*, *control_width=200*)

Frame similar to SimpleGUI *Frame* of CodeSkulptor.

_Frame__deal_event_joypad(*event*)
Private function that dispatch joypad *event*.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **event** – Pygame event

Returns True if some event match, else False

_Frame__deal_event_key(*event*)
Private function that dispatch key *event*.

Parameters **event** – Pygame event

Returns True if some event match, else False

_Frame__deal_event_mouse(*event*)
Private function that dispatch mouse *event*.

Parameters **event** – Pygame event

Returns True if some event match, else False

__init__(*title*, *canvas_width*, *canvas_height*, *control_width=200*)
Set the frame.

Don't use directly, use create_frame().

Parameters

- **title** – str
- **canvas_width** – (int or float) >= 0
- **canvas_height** – (int or float) >= 0
- **control_width** – (int or float) >= 0

__repr__()
Return '<Frame object>'.

Returns str

__weakref__
list of weak references to the object (if defined)

_background_pygame_color
Default background color of frame.

_canvas_border_pygame_color
Border color of canvas.

_controlpanel_background_pygame_color
Background color of control panel.

_cursor_auto_hide = False
When move cursor, if *True* then hide cursor when on canvas, else show cursor.

_cursor_in_canvas()

Returns *True* if the cursor is on canvas, *False* else.

_display_fps_average = False
If *True* then display FPS average on the canvas.

_draw_controlpanel()
Draw the control panel and two status boxes.

(Not available in SimpleGUI of CodeSkulptor.)

_draw_statuskey(*key=0*, *pressed=None*)
Draw the status box of key.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters

- **key** – int
- **pressed** – None or bool

_draw_statusmouse(*position=(0, 0)*, *pressed=None*)
Draw the status box of mouse.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters

- **position** – (int or float, int or float) or [int or float, int or float]
- **pressed** – bool

_fps = 60
Frames per second drawed (frequency of draw and check events)

_frame_instance = None
The only instance of Frame.

_frame_padding = 2
The padding in pixels around the canvas

_get_fps_average()
Return the framerate average (in frame per second) computed by Pygame.

(Not available in SimpleGUI of CodeSkulptor.)

Returns float

_hide_controlpanel = False
If *True* then hide control panel (and status box).

_hide_status = False
If *True* then hide status box.

_keep_timers = None
If *None* then ask (when stop frame) if it should be stop timers when program ending. (This is the default behavior.)

If *True* then timers keep running when program ending.

If *False* then stop all timers when program ending.

_pos_in_control(*x*, *y*)
If position (*x*, *y*) is on the zone of one *Control* or *TextAreaControl* then return it else return *None*.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters

- **x** – int or float
- **y** – int or float

Returns None or Control or TextAreaControl

_print_stats_cache = False
If *True* then print some statistics of caches after frame stopped.

_pygame_mode_depth = 0
Default number of bits used to represent color.

See https://www.pygame.org/docs/ref/display.html#pygame.display.set_mode

_pygame_mode_flags = 0
Default options of graphic mode.

See https://www.pygame.org/docs/ref/display.html#pygame.display.set_mode

classmethod _pygamecolors_cached_clear()
Empty the cache of Pygame colors used.

Each color used is cached to accelerate drawing. If you use many many different colors maybe use this function to free memory.

(Not available in SimpleGUI of CodeSkulptor.)

Side effect: Empty *_colors._PYGAMECOLORS_CACHED*.

classmethod _pygamefonts_cached_clear()
Empty the cache of Pygame fonts used.

Each font used with each size is cached to accelerate drawing. If you use many many different sizes maybe use this function to free memory.

(Not available in SimpleGUI of CodeSkulptor.)

Side effect: Empty *_fonts.__PYGAMEFONTS_CACHED*.

_save_canvas_and_stop(*filename*, *after=1000*)
Wait after ms (first wait until the frame is started), then save the canvas in a file and stop the program.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters

- **filename** – str
- **after** – int or float >= 0

_save_canvas_request(*filename*)
Request to save the canvas image in a file.

(The images are saved on each cycle fixed by *Frame._fps*.)

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **filename** – str

_save_canvas_requests = []
List of filenames in which to save canvas image.

_set_canvas_background_image(*image*)
Set an image to replace the background color of the canvas.

Parameters **image** – None or Image

classmethod _set_cursor_visible(*visible=True*)
If visible is *True* then show cursor, else hide cursor.

Independently of *_cursor_auto_hide* value.

Parameters **visible** – bool

_set_joypadaxe_handler(*joypad_handler*)
Set the function handler that will be executed (with the joypad index, the axe index and the value) when axis of joypad move.

(The events are checked on each cycle fixed by *Frame._fps*.)

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **joypad_handler** – function (int >= 0, int >=0, -1 <= float <= 1) -> *

_set_joypaddown_handler(*joypad_handler*)
Set the function handler that will be executed (with the joypad index and the button index) when a button of joypad is **pressed**.

(The events are checked on each cycle fixed by *Frame._fps*.)

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **joypad_handler** – function (int >= 0, int >= 0) -> *

_set_joypadhat_handler(*joypad_handler*)
Set the function handler that will be executed (with the joypad index, the hat index and the values (a, b) where a and b == -1, 0 or 1) when hat of joypad move.

(The events are checked on each cycle fixed by *Frame._fps*.)

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **joypad_handler** – function (int >= 0, int >= 0, (int, int)) -> *

_set_joypadup_handler(*joypad_handler*)
Set the function handler that will be executed (with the joypad index and the button index) when a button of joypad is **released**.

(The events are checked on each cycle fixed by *Frame._fps*.)

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **joypad_handler** – function (int >= 0, int >= 0) -> *

_statuskey_background_pygame_color
pygame.Color of background in status key box.

_statuskey_height = 20
Height of the status key box.

_statuskey_pygame_color
pygame.Color of status key box (text and rectangle).

_statuskey_pygame_font
pygame.font.Font of status key box.

_statusmouse_background_pygame_color
pygame.Color of background in status mouse box.

_statusmouse_height = 20
Height of the status mouse box.

_statusmouse_pygame_color
pygame.Color of status mouse box (text and rectangle).

_statusmouse_pygame_font
pygame.font.Font of status mouse box.

add_button(*text*, *button_handler*, *width=None*)
Add a button in the control panel.

When the button are pressed and released, *button_handler* are executed.

If *width* is not *None* then *text* is possibly cutted.

But, in CodeSkulptor, the accurate appearance is browser dependent. And in SimpleGUICS2Pygame, the accurate appearance is font dependent.

Parameters

- **text** – str
- **button_handler** – function () -> *
- **width** – None or int

Returns Control

add_input(*text*, *input_handler*, *width*)
Add a "label" with an input box in the control panel.

When click with left button of mouse on the "label" or input box, the focus is give to this input box.

When press Tab, the focus is give to the next input box (if exist).

When press Enter, this input box lost the focus and *input_handler* are executed with the input text.

Parameters

- **text** – str
- **input_handler** – function (str) -> *
- **width** – int

Returns Control

add_label(*text*, *width=None*)
Add a label in the control panel.

If *width* is not *None* then *text* is possibly cutted.

But, in CodeSkulptor, the accurate appearance is browser dependent. And in SimpleGUICS2Pygame, the accurate appearance is font dependent.

Parameters

- **text** – str
- **width** – None or int

Returns Control

download_canvas_image(*filename='canvas.png'*)
Save the content of the canvas in a local file.

In SimpleGUICS2Pygame supported formats are supported formats by Pygame to save: TGA, PNG, JPEG or BMP (see https://www.pygame.org/docs/ref/image.html#pygame.image.save).

If *filename* extension is not recognized then TGA format is used.

If *filename* == '' then a random filename is used, beginning by 'canvas_' and with '.png' extension.

In CodeSkulptor the format is always PNG.

(Available in SimpleGUI of CodeSkulptor3 but *not in CodeSkulptor 2* and *not in CodeSkulptor documentation*!)

Parameters **filename** – str

get_canvas_image()
NOT YET IMPLEMENTED! (Does nothing.)

(Available in SimpleGUI of CodeSkulptor but *not in CodeSkulptor documentation*!)

get_canvas_textwidth(*text*, *font_size*, *font_face='serif'*)
Return the width needed to draw *text* by *Frame.draw_text()*.

Parameters

- **text** – str
- **font_size** – (int or float) >= 0
- **font_face** – str == 'monospace', 'sans-serif', 'serif'

Returns int or float >= 0

set_canvas_background(*color*)
Set the background color of the canvas.

Parameters **color** – str

set_draw_handler(*draw_handler*)
Set the function handler that will be executed each cycle fixed by *Frame._fps*.

Parameters **draw_handler** – function (Canvas) -> *

set_keydown_handler(*key_handler*)
Set the function handler that will be executed (with the key code) when a key is released.

(The events are checked on each cycle fixed by *Frame._fps*.)

Parameters **key_handler** – function (int >= 0) -> *

set_keyup_handler(*key_handler*)
Set the function handler that will be executed (with the key code) when a key is pressed.

(The events are checked on each cycle fixed by *Frame._fps*.)

Parameters **key_handler** – function (int >= 0) -> *

set_mouseclick_handler(*mouse_handler*)
Set the function handler that will be executed (with the position of the mouse) when the left button of mouse is **released**.

(The events are checked on each cycle fixed by *Frame._fps*.)

Parameters **mouse_handler** – function ((int >= 0, int >= 0)) -> *

set_mousedrag_handler(*mouse_handler*)
Set the function handler that will be executed (with the position of the mouse) **for each** new mouse position when the left button of mouse is pressed.

(The events are checked on each cycle fixed by *Frame._fps*.)

Parameters **mouse_handler** – function ((int >= 0, int >= 0)) -> *

start()
Start the frame and these handler events.

> **Warning:** With SimpleGUICS2Pygame, `Frame.start()` is blocking until `Frame.stop()` execution or closing window. So timers must be started *before*, and states must be initialized *before*. (Or maybe after by a handler function.)

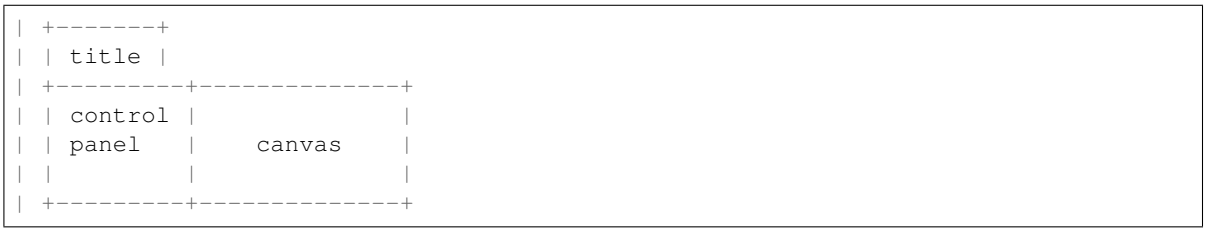
(In SimpleGUI of CodeSkulptor this function is *not* blocking.)

stop()
Stop frame activities.

If (Frame._keep_timers is None) and there is still running timers then ask in the canvas if it should be stop timers when program ending.

(Maybe available in SimpleGUI of CodeSkulptor but *not in CodeSkulptor documentation*!)

SimpleGUICS2Pygame.simpleguics2pygame.frame.**create_frame**(*title*, *canvas_width*, *canvas_height*, *control_width=200*)
Create and return an interactive window.



title: title of the window.
canvas_width, canvas_height: dimensions of the canvas.
control_width: width of the control panel.

(The frame is inactive until the execution of *Frame.start()*.)

Don't run twice!

Parameters

- **title** – str
- **canvas_width** – (int or float) >= 0
- **canvas_height** – (int or float) >= 0
- **control_width** – (int or float) >= 0

Returns Frame

SimpleGUICS2Pygame.simpleguics2pygame.frame.**__all__ = ('Frame', 'create_frame')**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

[source]

simpleguics2pygame — image

simpleguics2pygame module: simpleguics2pygame/image.

Class Image.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2015-2016, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version November 29, 2020

class SimpleGUICS2Pygame.simpleguics2pygame.image.**Image**(*url*)
Image similar to SimpleGUI *Image* of CodeSkulptor.

__init__(*url*)
Set an image.

Don't use directly, use *load_image()*.

Parameters **url** – str

__repr__()
Return '<*Image object*>'.

Returns str

__weakref__
list of weak references to the object (if defined)

_dir_search_first = '_img/'
load_image() try **first** to loading image from this directory, and next if failed, try to loading from URL.

This local directory is relative to the directory of your program.

_print_stats_cache(*text=''*, *short_url=True*)
Print to stderr some statistics of cached Pygame surfaces used by this image.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters

- **text** – str
- **short_url** – bool

_pygamesurfaces_cache_default_max_size = 1000
Default maximum number of Pygame surfaces in the *self._pygamesurfaces_cached*.

_pygamesurfaces_cached_clear()
Empty the cache of Pygame surfaces used by this image.

(Not available in SimpleGUI of CodeSkulptor.)

get_height()
Return the height ot this image.

(If initialization of this image was failed then return *0*.)

Returns int

get_width()
Return the width ot this image.

(If initialization of this image was failed then return *0*.)

Returns int

class SimpleGUICS2Pygame.simpleguics2pygame.image.**_LocalImage**(*filename*)
Child of Image to load local file image.

(Not available in SimpleGUI of CodeSkulptor.)

__init__(*filename*)
Set an image.

Don't use directly, use *_load_local_image()*.

Parameters **filename** – str

__repr__()
Return *'<_LocalImage object>'*.

Returns str

SimpleGUICS2Pygame.simpleguics2pygame.image.**load_image**(*url*)
Create and return an image by loading a file from *url*. Not founded URL and errors are ignored.

SimpleGUICS2Pygame try **first** to loading image from *Image._dir_search_first* local directory (*_img/* by default), and next if failed, try to loading from *url*.

This local directory is relative to the directory of your program.

For example, `load_image('http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/double_ship.png')` try first to loading from `_img/commondatastorage.googleapis.com/codeskulptor_assets/lathrop/double_ship.png`.

Supported formats are supported formats by Pygame to load: PNG, JPG, GIF (not animated)... (see https://www.pygame.org/docs/ref/image.html).

(CodeSkulptor may supported other formats, dependant on browser support.)

I recommend PNG and JPG format.

CodeSkulptor loads images **asynchronously** (the program continues without waiting for the images to be loaded). To handle this problem, you can use `simplegui_lib_loader.Loader` class.

Parameters **url** – str (**only a valid URL**, not local filename)

Returns Image

SimpleGUICS2Pygame.simpleguics2pygame.image.**_load_local_image**(*filename*)
Create and return an image by loading a file from *filename*. Not founded file and errors are ignored.

I recommend to use only Internet resources with the *load_image()* function. Then you can use your program **both** in standard Python and in CodeSkulptor. (See Tips.html#download-medias .)

But if it is necessary, you can load local image with this "private" function.

Supported formats are the same as the *load_image()* function.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **filename** – str (**only a valid filename**, not URL)

Returns _LocalImage

SimpleGUICS2Pygame.simpleguics2pygame.image.**__all__ = ('Image', '_LocalImage', 'load_image**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

[source]

simpleguics2pygame — keys

simpleguics2pygame module: simpleguics2pygame/keys.

Keys helpers.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2015, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version November 29, 2020

SimpleGUICS2Pygame.simpleguics2pygame.keys.**KEY_MAP = {'0': 48, '1': 49, '2': 50, '3': 5**
SimpleGUI keyboard characters contants.

simpleguics2pygame — sound

simpleguics2pygame module: simpleguics2pygame/sound.

Class Sound.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2015-2016, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version November 29, 2020

class SimpleGUICS2Pygame.simpleguics2pygame.sound.**_LocalSound**(*filename*)
Child of Sound to load local file sound.

(Not available in SimpleGUI of CodeSkulptor.)

__init__(*filename*)
Set a sound (if not Sound._load_disabled).

Don't use directly, use *_local_load_sound()*.

Parameters **filename** – str

__repr__()
Return *'<_LocalSound object>'*.

Returns str

class SimpleGUICS2Pygame.simpleguics2pygame.sound.**Sound**(*url*)
Sound similar to SimpleGUI *Sound* of CodeSkulptor.

__init__(*url*)
Set a sound (if not Sound._load_disabled).

Don't use directly, use *load_sound()*.

Parameters **url** – str

__repr__()
Return *'<Sound object>'*.

Returns str

__weakref__
list of weak references to the object (if defined)

_dir_search_first = '_snd/'
load_sound() try **first** to loading sound from this directory, and next if failed, try to loading from URL.

This local directory is relative to the directory of your program.

_get_length()
Return the length of this sound in seconds.

(If initialization of this sound was failed then return *0*.)

(Not available in SimpleGUI of CodeSkulptor.)

Returns int or float

_load_disabled = False
If *True* then load sounds are disabled.

pause()
Pause this sound. (Use *Sound.play()* to resume.)

play()
If this sound is paused then resume the sound, else start the sound.

rewind()
If this sound has already been started then stop the sound and rewind to the begining.

set_volume(*volume*)
Change the volume of this sound. The default volume is *1* (maximum).

Parameters **volume** – 0 <= int or float <= 1

SimpleGUICS2Pygame.simpleguics2pygame.sound.**create_sound**(*sound_data*, *sample_rate=8000*, *num_channels=1*)
NOT YET IMPLEMENTED! (Return an empty *Sound*.)

(Available in SimpleGUI of CodeSkulptor but *not in CodeSkulptor documentation*!)

Parameters

- **sound_data** – (tuple or list) of (0 <= int < 256)
- **sample_rate** – int >= 0
- **num_channels** – int >= 0

Returns Sound

SimpleGUICS2Pygame.simpleguics2pygame.sound.**load_sound**(*url*)
Create and return a sound by loading a file from *url*. Not founded URL and errors are ignored.

SimpleGUICS2Pygame try **first** to loading sound from *Sound._dir_search_first* local directory (*_snd/* by default), and next if failed, try to loading from *url*.

This local directory is relative to the directory of your program.

For example, `load_sound('http://commondatastorage.googleapis.com/codeskulptor-assets/jump.ogg')` try first to loading from `_snd/commondatastorage.googleapis.com/codeskulptor_assets/jump.ogg`.

Supported formats: OGG, WAV and MP3.

If MP3 sound failed on your system read installation of audioread.

(Supported formats by CodeSkulptor are browser dependant.)

(The sound can be started by *Sound.play()*.)

Parameters **url** – str (**only a valid URL**, not local filename)

Returns Sound

`SimpleGUICS2Pygame.simpleguics2pygame.sound.`**`_load_local_sound`**(*filename*)

Create and return a sound by loading a file from *filename*. Not founded file and errors are ignored.

I recommend to use only Internet resources with the *load_sound()* function. Then you can use your program **both** in standard Python and in CodeSkulptor. (See Tips.html#download-medias .)

But if it is necessary, you can load local sound with this "private" function.

Supported formats are the same as the *load_sound()* function.

(Not available in SimpleGUI of CodeSkulptor.)

Parameters **filename** – str (**only a valid filename**, not URL)

Returns _LocalSound

`SimpleGUICS2Pygame.simpleguics2pygame.sound.`**`__all__ = ('_LocalSound', 'Sound', 'create_sou`**

Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

[source]

simpleguics2pygame — timer

simpleguics2pygame module: simpleguics2pygame/timer.

Class Timer.

Don't require Pygame.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2015, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 20, 2020

`class` `SimpleGUICS2Pygame.simpleguics2pygame.timer.`**`Timer`**(*interval*, *timer_handler*)

Timer similar to SimpleGUI *Timer* of CodeSkulptor.

Don't require Pygame.

`__init__`(*interval*, *timer_handler*)

Set a time.

Don't use directly, use *create_timer()*.

Parameters

- **interval** – int or float > 0
- **timer_handler** – function () -> *

__repr__()
Return *'<Timer object>'*.

Returns str

__weakref__
list of weak references to the object (if defined)

classmethod _running_nb()
Return the number of running timers.

Returns int >= 0

classmethod _running_some()

Returns True if at least one timer running, else False

classmethod _stop_all()
Stop all timers.

(Not available in SimpleGUI of CodeSkulptor.)

Side effect: Empty *Timer._timers_running*.

_timers_running = {}
Dict {(Timer id): *Timer*} of all timers are running.

get_interval()
Return the interval of this timer.

(Maybe available in SimpleGUI of CodeSkulptor but *not in CodeSkulptor documentation*!)

Returns (int or float) > 0

is_running()
If this timer is running then return *True*, else return *False*.

Returns bool

start()
Start this timer.

> **Warning:** With SimpleGUICS2Pygame, `Frame.start()` is blocking until `Frame.stop()` execution or closing window. So timers must be started *before*, and states must be initialized *before*. (Or maybe after by a handler function.)

(Side effect: Add *id(self)*: *self* in *Timer._timers_running*.)

stop()
Stop this timer.

(Side effect: Remove *id(self)* of *Timer. _timers_running*.)

SimpleGUICS2Pygame.simpleguics2pygame.timer.**create_timer**(*interval*, *timer_handler*)
Create and return a timer that will execute the function *timer_handler* every *interval* milliseconds.

The first execution of *time_handler* will take place after the first period.

(The timer can be started by *Timer.start()*.)

Parameters

- **interval** – int or float > 0
- **timer_handler** – function () -> *

Returns Timer

SimpleGUICS2Pygame.simpleguics2pygame.timer.__all__ = ('Timer', 'create_timer')
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.

If the argument is a tuple, the return value is the same object.

[source]

Warning: Be careful, the main module *simpleguics2pygame* is split in several files, but items from these files are available from the *simpleguics2pygame* module itself, like the *simplegui* module of CodeSkulptor.

For example, the function *SimpleGUICS2Pygame.simpleguics2pygame.frame.create_frame()* must be used by *simplegui.create_frame()* in CodeSkulptor and also with SimpleGUICS2Pygame with *import SimpleGUICS2Pygame.simpleguics2pygame as simplegui*.

simpleguics2pygame module: simpleguics2pygame/__init__.

Standard Python (**2 and 3**) module reimplementing the SimpleGUI particular module of CodeSkulptor and CodeSkulptor3 (a Python browser environment).

Require Pygame (except for the Timer class).

Online HTML documentation on Read The Docs. (You can also see the online SimpleGUI documentation on CodeSkulptor or SimpleGUI documentation on CodeSkulptor3.)

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2016, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 19, 2020

- genindex
- modindex

8.2.6 simpleplot — replace the simpleplot module of CodeSkulptor

simpleplot module.

Replace the simpleplot module of CodeSkulptor.

Require matplotlib (and must be installed separately).

Warning: With SimpleGUICS2Pygame, if your program is terminated, then windows opened by `plot_bars()`, `plot_lines()` and `plot_scatter()` will be closed automatically. You can use the specific function `_block()` to block the program until closing all windows. See Tips to run specific code.

Piece of SimpleGUICS2Pygame. https://bitbucket.org/OPiMedia/simpleguics2pygame

license GPLv3 — Copyright (C) 2013-2016, 2020 Olivier Pirson

author Olivier Pirson — http://www.opimedia.be/

version May 20, 2020

SimpleGUICS2Pygame.simpleplot.**_COLORS = ('#edc240', '#afd8f8', '#cb4b4b', '#4da74d', '#944**
Color used for each graph.

(Not available in SimpleGUI of CodeSkulptor.)

SimpleGUICS2Pygame.simpleplot.**_MATPLOTLIB_AVAILABLE = True**
True if matplotlib is available, else *False*.

SimpleGUICS2Pygame.simpleplot.**_MATPLOTLIB_VERSION**
matplotlib.__version__ if Pygame is available, else *None*.

SimpleGUICS2Pygame.simpleplot.**_block**()
If some plot windows are open then block the program until closing all windows. **(Not available in SimpleGUI of CodeSkulptor.)**

SimpleGUICS2Pygame.simpleplot.**plot_bars**(*framename*, *width*, *height*, *xlabel*, *ylabel*, *datasets*, *legends=None*, *_block=False*, *_filename=None*)
Open a window titled *framename* and plot graphes with *datasets* data shown as vertical bars.

xlabel and *ylabel* are labels of x-axis and y-axis.

datasets must be a sequence of data. Each data must be:

- Sequence (not empty) of pair x, y. Each point (x, y) is represented by a vertical bar of height y.
- Or dict (not empty) x: y. Each point (x, y) is represented by a vertical bar of height y.

If *legends* is not None then it must be a sequence of legend of each graph.

If *_block* then block the program until closing the window else continue and close the window when program stop. **(Option not available in SimpleGUI of CodeSkulptor.)**

If *_filename* is not None then save the image to this file. **(Option not available in SimpleGUI of CodeSkulptor.)**

Parameters

- **framename** – str
- **width** – int > 0
- **height** – int > 0
- **xlabel** – str
- **ylabel** – str
- **datasets** – (list or tuple) of (((list or tuple) of ([int or float, int or float] or (int or float, int or float))) or (dict (int or float): (int or float)))
- **legends** – None or ((list or tuple) of same length as datasets)
- **_block** – False
- **_filename** – None or str

SimpleGUICS2Pygame.simpleplot.**plot_lines**(*framename*, *width*, *height*, *xlabel*, *ylabel*, *datasets*, *points=False*, *legends=None*, *_block=False*, *_filename=None*)
Open a window titled *framename* and plot graphes with *datasets* data shown as connected lines.

xlabel and *ylabel* are labels of x-axis and y-axis.

datasets must be a sequence of data. Each data must be:

- Sequence (not empty) of pair x, y. Each point (x, y) is plotted (in given order) and connected with line to previous and next points.
- Or dict (not empty) x: y. Each point (x, y) is plotted (in ascending order of x value) and connected with line to previous and next points.

If *points* then each point is highlighted by a small disc (a small circle in CodeSkulptor).

If *legends* is not None then it must be a sequence of legend of each graph.

If *_block* then block the program until closing the window else continue and close the window when program stop. **(Option not available in SimpleGUI of CodeSkulptor.)**

If *_filename* is not None then save the image to this file. **(Option not available in SimpleGUI of CodeSkulptor.)**

Parameters

- **framename** – str
- **width** – int > 0
- **height** – int > 0
- **xlabel** – str
- **ylabel** – str
- **datasets** – (list or tuple) of (((list or tuple) of ([int or float, int or float] or (int or float, int or float))) or (dict (int or float): (int or float)))
- **points** – bool
- **legends** – None or ((list or tuple) of same length as datasets)
- **_block** – False
- **_filename** – None or str

SimpleGUICS2Pygame.simpleplot.**plot_scatter**(*framename*, *width*, *height*, *xlabel*, *ylabel*, *datasets*, *legends=None*, *_block=False*, *_filename=None*)

Open a window titled *framename* and plot graphes with *datasets* data shown as scattered points.

xlabel and *ylabel* are labels of x-axis and y-axis.

datasets must be a sequence of data. Each data must be:

- Sequence (not empty) of pair x, y. Each point (x, y) is represented by a circle.
- Or dict (not empty) x: y. Each point (x, y) is represented by a circle.

If *legends* is not None then it must be a sequence of legend of each graph.

If *_block* then block the program until closing the window else continue and close the window when program stop. **(Option not available in SimpleGUI of CodeSkulptor.)**

If *_filename* is not None then save the image to this file. **(Option not available in SimpleGUI of CodeSkulptor.)**

Parameters

- **framename** – str
- **width** – int > 0
- **height** – int > 0

- **xlabel** – str
- **ylabel** – str
- **datasets** – (list or tuple) of (((list or tuple) of ([int or float, int or float] or (int or float, int or float))) or (dict (int or float): (int or float)))
- **legends** – None or ((list or tuple) of same length as datasets)
- **_block** – False
- **_filename** – None or str

[source]

Warning: Be careful, the main module *simpleguics2pygame* is split in several files, but items from these files are available from the *simpleguics2pygame* module itself, like the *simplegui* module of CodeSkulptor.

For example, the function *SimpleGUICS2Pygame.simpleguics2pygame.frame.create_frame()* must be used by *simplegui.create_frame()* in CodeSkulptor and also with SimpleGUICS2Pygame with *import SimpleGUICS2Pygame.simpleguics2pygame as simplegui*.

- genindex
- modindex

8.3 Compatibility

8.3.1 Compatibility between SimpleGUI of CodeSkulptor and SimpleGUICS2Pygame

- With SimpleGUI of CodeSkulptor or CodeSkulptor3, some things (like fonts, buttons...) are browser dependents. That is to say that there are some differences from one browser to another. Therefore it is normal that these elements are also slightly different with SimpleGUICS2Pygame.
- With SimpleGUI of CodeSkulptor or CodeSkulptor3, supported sound formats (WAV, OGG, MP3) are also browser dependents.

  With SimpleGUICS2Pygame WAV, OGG and MP3 (since version 2.0.0) are supported.

  I recommend to always use the OGG format.

- **Warning:** With SimpleGUICS2Pygame, `Frame.start()` is blocking until `Frame.stop()` execution or closing window. So timers must be started *before*, and states must be initialized *before*. (Or maybe after by a handler function.)

- **Warning:** With SimpleGUICS2Pygame, if your program is terminated, then windows opened by `plot_bars()`, `plot_lines()` and `plot_scatter()` will be closed automatically. You can use the specific function `_block()` to block the program until closing all windows.

- See Tips to run specific code.

Problems with SimpleGUICS2Pygame

If MP3 sound failed on your system read installation of audioread.

8.3.2 Compatibility between Python 2 and Python 3

CodeSkulptor implements a subset of Python 2.

CodeSkulptor3 implements a subset of Python 3.

You can use SimpleGUICS2Pygame with Python 2 and Python 3.

- The division `/` don't have the same behavior in Python 2 and Python 3: https://docs.python.org/3/whatsnew/3.0.html#integers .
  - In Python 2 (and CodeSkulptor): `3/2 == 1` and `3/2.0 == 1.5`
  - In Python 3: `3/2 == 3/2.0 == 1.5`

  `3//2 == 1` and `3//2.0 == 1.0` **everywehre**.

  (You can add `from __future__ import division` *on the top* of your program, and Python 2 mimic Python 3 division. But then *CodeSkulptor failed*!)
- Rounded behavior is also different: https://docs.python.org/3/whatsnew/3.0.html#builtins .
  - In Python 2 (and CodeSkulptor): `round(1.5) == 2.0` and `round(2.5) == 3.0`
  - In Python 3: `round(1.5) == round(2.5) == 2`
- `print` is a function in Python 3: https://docs.python.org/3/whatsnew/3.0.html#print-is-a-function .
  - In Python 2 (and CodeSkulptor): `print 'Hello real world!', 42`
  - In Python 3: `print('Hello real world!', 42)`

  With only one argument, `print('Hello real world! ' + str(42))` run **everywhere**.

  (You can add `from __future__ import print_function` *on the top* of your program, and Python 2 mimic Python 3 print function. But then *CodeSkulptor failed*!)

8.3.3 Compatibility between CodeSkulptor and CodeSkulptor3

CodeSkulptor implements a subset of Python 2.

CodeSkulptor3 implements a subset of Python 3.

There are moreover some other differences.

- In `simplegui module` module of CodeSkulptor3 and SimpleGUICS2Pygame, there is a `Canvas.draw_arc()` method that doesn't exist in CodeSkulptor2.
- In `simplegui module` module of CodeSkulptor3 and SimpleGUICS2Pygame, there is a `Frame.download_canvas_image()` method that doesn't exist in CodeSkulptor2.
- In `numeric` module of CodeSkulptor 2 and SimpleGUICS2Pygame, there is a `Matric.scale()` method to multiply the matrix by a scalar and the operator `*` multiply two matrices.

  In CodeSkulptor3, the `Matric.scale()` method doesn't exist, the operator `*` multiply a matrix by a scalar and the operator `@` multiply two matrices.

Problem with CodeSkulptor

If you have no sound with CodeSkulptor on Chrome/Chromium and GNU/Linux, maybe you can fix that problem with `pavucontrol`. For example on Debian, install it and set the volume for the application:

```
$ sudo apt install pavucontrol
$ pavucontrol
```

8.3.4 Comparison of speeds

Due to the execution in browser CodeSkulptor is slower than directly execute Python on your computer. Below a graph of speeds of the execution of the test program example/Stress_Balls/Stress_Balls.py in several environments (all on Intel Xeon W3530 Quad-Core 2.8 GHz 6 Gio), where the canvas is *normally* displayed 60 times by second (SimpleGUITk seems not very precise). When there is a lot of shapes to draw the execution can no longer finished all drawing during the given time.

`-O` indicates that assertions was disabled during the test. See Python assertions option section.

You can also execute this online program to plot results: example/Stress_Balls/Stress_Balls_results.py.

8.4 Tips

8.4.1 CodeSkulptor

CodeSkulptor is a Python implementation (in JavaScript) running in a browser. It implements a subset of Python 2.

CodeSkulptor3 is the same but it implements a subset of Python 3.

It is the environment used in the MOOC An Introduction to Interactive Programming in Python (Rice University, Coursera).

To use a program from CodeSkulptor in *standard Python* (with this package), you need to change `import simplegui` by `import SimpleGUICS2Pygame.simpleguics2pygame as simplegui`.

The right way to do is to write this

```
try:
    import simplegui
except ImportError:
    import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
```

and your program **runs both** in CodeSkulptor and *standard Python*.

So, if your program runs in CodeSkulptor, it imports `simplegui`. Else, an `ImportError` exception will be raised, and then it will imports `SimpleGUICS2Pygame.simpleguics2pygame` and it renamed to `simplegui`.

In this package a little script `cs2both.py` can help to quickly make this changement on program downloaded from CodeSkulptor.

Run `python cs2both.py YOURPROGRAM.py`.

The file `YOURPROGRAM.py` is copied to `YOURPROGRAM.py.bak` before changing.

To use also the **other modules**, you can write this. But specify only those you use.

```
try:
    import simplegui

    import codeskulptor
    import numeric
    import simpleplot

    import user305_SXBsmszNiUxIeoV as codeskulptor_lib
    import user305_SZNWcbqQHXN4pow as simplegui_lib
except ImportError:
    import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

    import SimpleGUICS2Pygame.codeskulptor as codeskulptor
    import SimpleGUICS2Pygame.numeric as numeric
    import SimpleGUICS2Pygame.simpleplot as simpleplot

    import SimpleGUICS2Pygame.codeskulptor_lib as codeskulptor_lib
    import SimpleGUICS2Pygame.simplegui_lib as simplegui_lib
```

Note that import name like `user305_SXBsmszNiUxIeoV` in CodeSkulptor is valid both for CodeSkulptor 2 and CodeSkulptor3. It corresponds to URLs http://www.codeskulptor.org/#user305_SXBsmszNiUxIeoV.py and https://py3.codeskulptor.org/#user305_SXBsmszNiUxIeoV.py .

8.4.2 Specific code

To run specific code on CodeSkulptor or with SimpleGUICS2Pygame, you can write this

```
try:
    import simplegui

    SIMPLEGUICS2PYGAME = False
except ImportError:
    import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

    SIMPLEGUICS2PYGAME = True
```

And then you can run specific code simply by testing value of `SIMPLEGUICS2PYGAME`. For example:

```
# ...

def joypad_up(joypad, button):
    if joypad == 0:
        if button == 0:
            # ...

if SIMPLEGUICS2PYGAME:
    frame._set_joypadup_handler(joypad_up)
```

8.4.3 Joypads

SimpleGUICS2Pygame adds the possibility to use joypads. It is *not* available in CodeSkulptor. You can make compatible program with the previous tip to separate specific code.

Similarly to `set_mouseclick_handler()` and `set_mousedrag_handler()` functions, the class Frame in SimpleGUICS2Pygame defines `_set_joypaddown_handler()`, `_set_joypadup_handler()`, `_set_joypadaxe_handler()` and `_set_joypadhat_handler()`.

Two little examples using joypads (but of course only when you run them with SimpleGUICS2Pygame directly on your computer): example/Pong.py and example/RiceRocks_Asteroids.py.

8.4.4 Colors

The color parameter used by drawing functions must be in the following formats:

- `'#rrggbb'` with rr, gg, bb hexadecimal numbers on 2 figures
- `'#rgb'` with r, g, b hexadecimal numbers on 1 figure
- `'rbg(red,blue,green)'` with red, blue, green 0 <= integer <= 255
- `'rgba(red,blue,green,alpha)'` with red, blue, green 0 <= integer <= 255 and alpha between 0 and 1
- a constant name in this list https://www.w3schools.com/colors/colors_names.asp .

See the official HTML colors: http://www.opimedia.be/DS/mementos/colors.htm .

8.4.5 Command line arguments

When you run a program you can use following arguments: `python YOURPROGRAM.py [SimpleGUICS2Pygame arguments] [application arguments]`

- `--default-font`: Use Pygame default font instead serif, monospace... (this is faster if you display a lot of text).
- `--display-fps`: Display FPS average on the canvas.
- `--fps N`: Set Frame Per Second (default: 60 FPS).
- `--frame-padding N`: Set the padding in pixels found around the canvas (default: 2).
- `--fullscreen`: Fullscreen mode.
- `--help`: Print help message and quit.
- `--keep-timers`: Keep running timers when close frame without asking (default: ask before close). See also `--stop-timers`.
- `--last`: Mark this argument as the last SimpleGUICS2Pygame's argument. (Do nothing else.)
- `--no-border`: Window without border.
- `--no-controlpanel`: Hide the control panel (and status boxes).
- `--no-load-sound`: Don't load any sound.
- `--no-status`: Hide two status boxes.
- `--overwrite-downloaded-medias`: Download all images and sounds from Web and save in local directory even if they already exist.
- `--print-application-args`: Print remaining arguments transmit to application.

- `--print-args`: Print final configuration from SimpleGUICS2Pygame's argument.
- `--print-load-medias`: Print URLs or local filenames loaded.
- `--print-stats-cache`: After frame stopped, print some statistics of caches.
- `--save-downloaded-medias`: Save images and sounds downloaded from Web that don't already exist in local directory.
- `--stop-timers`: Stop all timers when ending program (default: running timers continue, as in CodeSkulptor). See also `--keep-timers`.
- `--version`: Print help message and quit.

If an argument is not in this list then it is ignored and all next arguments are ignored by SimpleGUICS2Pygame.

Arguments used by SimpleGUICS2Pygame is deleted to `sys.argv`. Remaining arguments can used by your application.

SimpleGUICS2Pygame arguments are automatically read when the module `simpleguics2pygame` is imported.

Examples:

- `python YOURPROGRAM.py --no-controlpanel --stop-timers --foo --fullscreen`
  Run `YOURPROGRAM.py` with the control panel hidden and timers will stoped. But SimpleGUICS2Pygame ignore `--foo` and `--fullscreen`.
  `YOURPROGRAM.py` application receive `--foo --fullscreen` arguments.
- `python YOURPROGRAM.py --no-controlpanel --last --stop-timers --foo --fps 30`
  Run `YOURPROGRAM.py` with the control panel hidden. But SimpleGUICS2Pygame ignore `--stop-timers`, `--foo`, `--fps` and `30`.
  `YOURPROGRAM.py` application receive `--stop-timers --foo --fps 30` arguments.

8.4.6 Download medias

Run `python YOURPROGRAM.py --save-downloaded-medias --print-load-medias` once. Images and sounds used (from URLs) will be saved in local directory (`_img/` et `_snd/` by default). Next simply run `python YOURPROGRAM.py` and the medias will be loaded from these local directories.

For example, `load_image('http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/double_ship.png')` save image to `_img/commondatastorage.googleapis.com/codeskulptor_assets/lathrop/double_ship.png`.

SimpleGUICS2Pygame has two additional classes to load directly local files: _LocalImage() and _LocalSound(). But be aware that these functions are *not* available in CodeSkulptor.

8.4.7 Helper functions

This package contains 5 additional modules with several helper functions that you can also import online in CodeSkulptor:

- codeskulptor_lib — some miscellaneous functions
- simplegui_lib_draw — draw functions
- simplegui_lib_fps — class to calculate and display Frames Per Second
- simplegui_lib_keys — class to manage keyboard handling

- simplegui_lib_loader — class to load images and sounds

For example, to draw multiline text you can use draw_text_multi() from the simplegui_lib_draw module by:

```
try:
    import simplegui

    import user305_SaT1YKoOikl4ax9 as simplegui_lib_draw
except ImportError:
    import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

    import SimpleGUICS2Pygame.simplegui_lib as simplegui_lib_draw

def draw(canvas):
    # ...
    draw_text_multi(canvas,
                    """line 1
line 2
line 3""", (x, y), size, 'white', 'serif')
    # ...
```

8.4.8 Python assertions option

Run `python YOURPROGRAM.py` then asserts is enabled and this package is (intentionnaly) very strict. So maybe "correct" programs in CodeSkulptor fail! (In fact CodeSkulptor is very permissive. Some incorrect Python codes are accepted.) It is a good point to develop and write *correct programs*. But if you want just run a program without annoying assertions you can *disable* them with. `python -O YOURPROGRAM.py`.

In some cases run without assertions is **faster**. See in the Comparison of speeds section, an example where SimpleGUICS2Pygame functions are executed a lot of times.

8.4.9 Ressources: images, sounds and example programs

Online images & sounds links

Python programs running in CodeSkulptor

8.5 License: GPLv3

8.5.1 Author: Olivier Pirson — OPi 

Website: http://www.opimedia.be/

Bitbucket: https://bitbucket.org/OPiMedia/

- olivier.pirson.opi@gmail.com
- Mastodon: https://mamot.fr/@OPiMedia — Twitter: https://twitter.com/OPirson
- diaspora* (Framasphere*): https://framasphere.org/u/opimedia
- LinkedIn: https://www.linkedin.com/in/olivierpirson/ — CV: http://www.opimedia.be/CV/English.html
- other profiles: http://www.opimedia.be/about/

8.5.2 Support me

This program is a **free software** (GPL license). It is **completely free** (like "free speech" *and* like "free beer"). However you can **support me** financially by donating.

Click to this link 

Thank you!

8.6 Known bugs

8.6.1 Old bug

- On my old GNU/Linux Debian 7 *Wheezy*, sometime the Pygame mixer initialization blocked the exit of the program when another application play also sounds.

  With my current GNU/Linux Debian 8 *Jessie*, I never had this problem.

8.7 Developers

This is a **free software**, so you can download it, **modify it** and **submit your modifications**. You can also **redistribute** your own version (keeping the GPL license).

Complete **sources** on Bitbucket: https://bitbucket.org/OPiMedia/simpleguics2pygame

8.7.1 Naming convention

Usually in Python, prepending a name by underscore _ is a convention for non-public items. Here intention is to separate additional variables and functions in SimpleGUICS2Pygame to the standard API of CodeSkulptor. One goal of SimpleGUICS2Pygame is to mimic as much as possible the behavior of CodeSkulptor. Additional functionalities are isolated with **prepending underscore** _.

8.7.2 Hierarchy of files on Bitbucket

- SimpleGUICS2Pygame/: **source** code
  - example/: little example programs and games
  - script/
    - cs2both.py: Python program to modify automatically your CodeSkulptor programs to run with SimpleGUICS2Pygame
    - pygame_check.py: Python program to check Pygame installation
    - SimpleGUICS2Pygame_check.py: Python program to **check installation**
  - simpleguics2pygame/: **main module** (splitted in several files) that reimplementing the `simpleguics2pygame` module of CodeSkulptor
  - test/: **test files**, mainly to check compatibility with CodeSkulptor
  - codeskulptor.py: module that reimplementing the `codeskulptor` module of CodeSkulptor
  - numeric.py: module that reimplementing the `numeric` module of CodeSkulptor
  - simpleplot.py: module that reimplementing the `simpleplot` module of CodeSkulptor
  - ...
- Sphinx/: source **documentation**
  - _static/links/: links of programs, images and sounds
  - ...
- stuffs/: unimportant stuff
- _dist/: last and previous versions of installation **archive files**
- _img/: logos
- Makefile: to build documentation, distributions, etc.
- setup.py: Python installation file
- ...

Warning: Before the version 2.0.0, the main module `simpleguics2pygame` was one file. Now it is splitted in several files in `simpleguics2pygame/` subdirectory.

8.7.3 Diagrams of imports

Auto-generated by pydeps.

Only files of SimpleGUICS2Pygame.simpleguics2pygame package

All files

8.7.4 Class diagrams

Auto-generated by Pyreverse.

Only public members

With private members

8.8 ChangeLog

- 2.1.0 — November 29, 2020
  - Removed Pygame restriction to version 1.9.6.
  - Removed old special cases when Pygame was not installed.
  - Cleaned some warnings from mypy.
  - `example/Nostalgic_Basic_Blitz.py`: corrected bug with negative position of bomb.
- 2.0.3 — October 2, 2020
  - Corrected reading permission of files in distribution files.
  - Added Arch Linux installation script (written by Danny Fajardo).
- 2.0.2 — May 23, 2020
  - Documentation:
    - Updated image and sound links.
  - Tests:
    - Completed missing type annotations in simpleguics2pygame/control.
- 2.0.1 — May 21, 2020
  - Documentation:
    - Added class diagram generated by Pyreverse.
    - Updated image and CodeSkulptor program links.
  - Program examples:
    - Adapted `example/Memory.py` with images moved to HTTPS.
  - Tests:
    - Added type annotations (in Python 2 mode) for each function.
    - Cleaned some type annotations instead ignore them.
- 2.0.0 — April 18, 2020
  - Converted from Mercurial version control system to Git.
  - Corrected files included in MANIFEST.in for distribution building. (Thanks to *7coil*.)

- Improved installation. Now all **requirements are automatically installed**.
- Modules:
  - **Splitted the big file ‘‘simpleguics2pygame.py‘‘.**
  - Added alpha possibility on background color.
  - Added dealing of **joypads**.
  - Added dealing of **MP3** sounds. Added `draw_arc()` in `Canvas` and `test_arc`.
  - Added `Frame.download_canvas_image()`, `Frame._cursor_auto_hide` and `Frame._set_cursor_visible()`.
  - Added `codeskulptor_version()` in `codeskulptor_lib`.
  - Added `draw_text_multi()` in `simplegui_lib_draw`.
  - Added `--frame-padding` (thanks to *7coil*), `--last`, `--help`, `--print-application-args`, `--print-args` and `--version` command line options.
  - Added `randomize_iteration()` in `codeskulptor`.
  - Added `transparent` “color” name.
  - Added `ValueError` exception if `draw_text()` try to draw a text containing unprintable whitespace character.
  - Corrected `keys` parameter use in `simplegui_lib_keys.Keys()`.
  - Improved dealing of input box.
  - Updated `simpleplot` module, to “run” same if matplotlib is not installed.
- Documentation:
  - Corrected “Read the Docs” subpackage problem.
  - Added a developer’s page.
  - Replaced `_WEBSITE` value by documentation link.
  - Updated. (Thanks to *John Gray* and *Tom Keller*.)
  - Splitted media links to image links and sound links.
  - Updated installation documentation.
  - Updated media and CodeSkulptor programs links.
- Program examples:
  - Added `example/presentation.py`.
  - Added `example/stop_example.py`.
  - Moved from CodeSkulptor to CodeSkulptor3.
- Scripts:
  - Added `script/pygame_check.py` to check Pygame installation alone.
  - Updated `script/SimpleGUICS2Pygame_check.py`.
- Tests:
  - Added static checking in `Makefile`, and corrected a lot of style warnings.
  - Corrected and updated `test/test_sound.py`.

    * Added `test/test_command_line_options.py`.
    * Added `test/test_input.py`.
    * Updated `test/test_dir.py`.
    * Updated `test/test_objects.py`.
    * Updated `test/test_text.py`.
- 01.09.00 — January 1st, 2015
  - **Added ‘‘_load_local_image()‘‘ and ‘‘_load_local_sound()‘‘ functions.**
  - Added `test/test_sound.py`.
  - Updated `test/test_dir.py`.
  - Updated `test/test_image.py`.
  - Added `--fps n` option.
  - Added Donate button in `_draw_about()` panel.
- 01.08.01 — October 9, 2014
  - Added information when pygame is not installed.
  - Corrected local filename bug in `_load_media()`. (Thanks to Sergey Sorokin.)
  - Updated documentation.
- 01.08.00 — October 4, 2014
  - Added alternative grey colors.
  - Added HSL and HSLA colors format.
  - Added `test/test_colors_html_hsla.py`.
  - Updated CodeSkulptor programs links.
  - Updated `codeskulptor_lib`.
  - Updated `test/test_colors_html_rgba.py`.
  - Updated media links.
- 01.07.00 — September 2, 2014
  - Added `plot_scatter()` function in `simpleplot` module.
  - Added `test/test_simpleplot_scatter.py`.
  - Updated `test/test_dir.py`.
  - Updated documentation.
  - Updated CodeSkulptor programs links.
- 01.06.03 — July 24, 2014
  - Implemented `width` parameter in `add_label()`.
  - Added `test/test_button_label.py`.
- 01.06.02 — July 18, 2014
  - Corrected stupid error in `add_label()`.
- 01.06.01 — July 17, 2014

  - Added (fake) width parameter in `add_label()`.
  - Corrected gz archive of HTML offline documentation.
  - Added private members in all documentation.
- 01.06.00 — June 16, 2014
  - Updated `numeric`.
  - Updated `example/Spaceship_prototype.py` and `example/RiceRocks_Asteroids.py`.
  - Updated `test/test_dir.py`.
  - Added `Loader.cache_clear()` and `Loader.print_stats_cache()`.
  - Added a cache mechanism to Pygame surfaces used by `Image` (**improve speed** of `draw_image()`).
  - Added `Image._url` attribute.
  - Moved `_RADIAN_TO_DEGREE`.
  - Print now to `stderr` instead `stdout`.
  - Updated `_draw_about()`.
  - Updated documentation.
  - Updated media and CodeSkulptor programs links.
- 01.05.00 — May 25, 2014
  - Added cache for colors and option `--print-stats-cache`.
  - First public version of `.hgignore` and `Makefile`.
  - Off the mixer if no sound is loaded.
  - Updated `example/RiceRocks_Asteroids.py`.
  - Updated documentation.
  - Updated `example/Spaceship_prototype.py`.
  - Updated `example/Blackjack.py`.
  - Updated `example/Memory.py`.
  - Updated `example/Pong.py`.
  - Cosmetic changes in some example programs.
  - Updated `test/test_all.py`.
  - Better order Pygame initalization.
  - Updated `script/cs2both.py` and `script/SimpleGUICS2Pygame_check.py`.
  - Updated `simplegui_lib_keys.py` and `example/keys.py`.
  - Updated `example/Stopwatch.py`.
  - Changed filename used by `_load_media()` (use now the query part of URLs).
  - Added precision to Window$ installation.
  - Updated media and CodeSkulptor programs links.
- 01.04.00 — December 16, 2013
  - Customized documentation.

  - Splitted changes in a separated file.
  - Added `numeric` (`Matrix` object) module.
  - Corrected some typos by Maxim Rybalov. (Thank you.)
  - Updated `simplegui_lib_fps.py`.
  - Updated `example/RiceRocks_Asteroids.py`.
- 01.03.00 — December 13, 2013
  - Removed exception to `get_canvas_image()`.
  - Updated almost all files to add `except ImportError`.
  - Updated `codeskulptor_lib.codeskulptor_is()`.
  - Added `simplegui_lib_fps.py`.
  - Corrected bug in `_load_media()` (issue #1). (Thanks to Sean Flanigan.)
  - Updated documentation to clarify local use of images and sounds. (Thanks to Ines Simicic.)
  - Updated `script/cs2both.py`.
  - Corrected conversion of `_fps_average` to `int` in Python 2.
  - Corrected mentions of `Frame._fps` in comment.
  - Updated `example/Blackjack.py`.
  - Updated `example/Spaceship_prototype.py`.
  - Updated `example/Memory.py`.
  - Updated media and CodeSkulptor programs links.
- 01.02.00 — November 8, 2013
  - Splitted `simplegui_lib.py` in `simplegui_lib.py`, `simplegui_lib_draw.py` and `simplegui_lib_loader.py`.
  - Added `simplegui_lib_keys.py`.
  - Added `example/keys.py` and `example/loader.py`.
  - Updated `example/RiceRocks_Asteroids.py` and `example/Spaceship_prototype.py`.
  - Updated `script/SimpleGUICS2Pygame_check.py`.
  - Updated `test/test_image.py` and `test/test_text.py`.
  - Updated media and CodeSkulptor programs links.
  - Corrected installation documentation.
- 01.01.00 — November 1st, 2013
  - Added `_block` and `_filename` parameters in `simpleplot.plot_lines()` function.
  - Added `plot_bars()` function in `simpleplot` module.
  - Added `test/test_simpleplot_bars.py` and `test/test_simpleplot_lines.py`.
  - Updated `test/test_all.py`.
  - Updated media links.
  - Corrected minor errors in documentation.

  - Added `set_timeout()` function in `codeskulptor` module.
  - Updated `example/Mandelbrot_Set.py` (used `set_timeout()`).
  - Updated CodeSkulptor programs links.
- 01.00.02 — October 31, 2013
  - Corrected bug in `TextAreaControl.set_text()`: the label text was also modified.
  - Updated documentation.
  - Updated `cs2both.py`.
  - Updated `example/Mandelbrot_Set.py` (optimized draw).
  - Updated media and CodeSkulptor programs links.
- 01.00.01 — October 9, 2013
  - Adapted documentation and `cs2both.py` to changes of CodeSkulptor (`int` and `float` are now separate).
- 01.00.00 — July 13, 2013
  - Moved documentation to Read The Docs.
  - Added `simpleplot` module.
  - Updated `example/Mandelbrot_Set.py` (used vertical symetry).
  - Updated media and CodeSkulptor programs links.
- 00.92.00 — June 27, 2013
  - Changed `simplegui_lib.Loader` class to display progression loading in SimpleGUICS2Pygame (moved arguments from `wait_loaded()` function to `__init__()`).
  - Replaced `Frame._already_frame` by `Frame._frame_instance`.
  - Updated `example/RiceRocks_Asteroids.py` (collisions of asteroids and little asteroids).
  - Added `Frame._set_canvas_background_image()` function.
  - Memoization of downloaded images and sounds.
  - Changed save in local directory to avoid conflict.
  - Added `test/test_image.py`.
  - Added `--overwrite-downloaded-medias` and `--save-downloaded-medias` options.
  - Display versions in `script/SimpleGUICS2Pygame_check.py`.
- 00.91.00 — June 23, 2013
  - Changed installation program to build distributions (now `setuptools` is used).
  - Added `--print-load-medias` option.
  - Added `script/SimpleGUICS2Pygame_check.py` and moved and updated `cs2both.py`.
  - Now, `_set_option_from_argv()` deleted SimpleGUICS2Pygame options after use.
  - Memoization of Pygame fonts.
  - Added `--default-font` option.
  - Many cosmetic changes to respect PEP 8.
  - Updated media and CodeSkulptor programs links.

  - Some precisions and English corrections in the documentation.
  - Added some CodeSkulptor programs links.
  - `example/Memory.py`: moved image locations.
  - `example/Nostalgic_Basic_Blitz.py`: added spacebar information.
- 00.90.10 — June 19, 2013
  - Adapted button, label and input to display multine text.
  - Simplified handler functions transmitted to `add_button()` in some programs.
  - Added `example/Nostalgic_Basic_Blitz.py`.
  - Changed `default_pygame_color` param of `_simpleguicolor_to_pygamecolor()` function (now installation is ok even if Pygame not installed).
  - Moved `_VERSION` and `_WEBSITE` constants from `simpleguics2pygame.py` to `__init__.py`.
  - Removed `enumerate()` function from `codeskulptor_lib` (now implemented natively by CodeSkulptor).
  - Added `--display-fps` option.
  - Added `example/RiceRocks_Asteroids.py`.
  - Updated some CodeSkulptor programs links.
  - Added some new media links.
  - Added some details in documentations.
  - Some cosmetic changes.
- 00.90.00 — June 13, 2013
  - First public version.

CHAPTER 9

Indices and tables

- genindex
- modindex

Python Module Index

S

Index

Symbols

A

C

D

F

G

H

I

K

L

M

P

R

S

T

W