Additional WxPython widgets

The classes in this module are based on internal classes that where originally designed to help the creation of graphical tools for the control and the visualization of audio signals. WxPython must be installed under the current Python distribution to access these classes.

PyoGuiControlSlider

class **PyoGuiControlSlider**(*parent*, *minvalue*, *maxvalue*, *init=None*, *pos=(0, 0)*, *size=(200, 16)*, *log=False*, *integer=False*, *powoftwo=False*, *orient=4*) [source]

Floating-point control slider.

Parent: wx.Panel

Events: EVT_PYO_GUI_CONTROL_SLIDER

Sent after any change of the slider position. The current value of the slider can be retrieve with the *value* attribute of the generated event. The object itself can be retrieve with the *object* attribute of the event and the object's id with the *id* attrbute.

Args: parent: wx.Window

The parent window.

minvalue: float

The minimum value of the slider.

maxvalue: float

The maximum value of the slider.

init: float, optional

The initial value of the slider. If None, the slider inits to the minimum value. Defaults to None.

pos: tuple, optional

The slider's position in pixel (x, y). Defaults to (0, 0).

size: tuple, optional

The slider's size in pixel (x, y). Defaults to (200, 16).

log: boolean, optional

If True, creates a logarithmic slider (minvalue must be greater than 0). Defaults to False.

integer: boolean, optional

If True, creates an integer slider. Defaults to False.

powoftwo: boolean, optional

If True, creates a power-of-two slider (log is automatically False and integer is True). If True, minvalue and maxvalue must be exponents to base 2 but init is a real power-

of-two value. Defaults to False.

orient: {wx.HORIZONTAL or wx.VERTICAL}, optional
The slider's orientation. Defaults to wx.HORIZONTAL.

enable() [source]

Enable the slider for user input.

disable() [source]

Disable the slider for user input.

setValue(*x*, *propagate=True*) [source]

Sets a new value to the slider.

Args: x: int or float
The controller number.

propagate: boolean, optional
If True, an event will be sent after the call.

setMidiCtl(*x*, *propagate=True*) [source]

Sets the midi controller number to show on the slider.

Args: x: int
The controller number.

propagate: boolean, optional
If True, an event will be sent after the call.

setRange(*minvalue*, *maxvalue*) [source]

Sets new minimum and maximum values.

Args: minvalue: int or float
The new minimum value.

maxvalue: int or float
The new maximum value.

getValue() [source]

Returns the current value of the slider.

getMidiCtl() [source]

Returns the midi controller number, if any, assigned to the slider.

getMinValue() [source]

Returns the current minimum value.

getMaxValue() [source]

Returns the current maximum value.

getInit() [source]

Returns the initial value.

getRange() [source]

Returns minimum and maximum values as a list.

isInteger() [source]

Returns True if the slider manage only integer, False otherwise.

isLog() [source]

Returns True if the slider is logarithmic, False otherwise.

isPowOfTwo() [source]

Returns True if the slider manage only power-of-two values, False otherwise.

PyoGuiVuMeter

class **PyoGuiVuMeter**(*parent*, *nchnls=2*, *pos=(0, 0)*, *size=(200, 11)*, *orient=4*, *style=0*) [source]

Multi-channels Vu Meter.

When registered as the Server's meter, its internal method *setRms* will be called each buffer size with a list of normalized amplitudes as argument. The *setRms* method can also be registered as the function callback of a PeakAmp object.

Parent: wx.Panel

Args: parent: wx.Window

The parent window.

nchnls: int, optional

The initial number of channels of the meter. Defaults to 2.

pos: wx.Point, optional

Window position in pixels. Defaults to (0, 0).

size: tuple, optional

The meter's size in pixels (x, y). Defaults to (200, 11).

orient: {wx.HORIZONTAL or wx.VERTICAL}, optional

The meter's orientation. Defaults to wx.HORIZONTAL.

style: int, optional

Window style (see wx.Window documentation). Defaults to 0.

setNchnls(*nchnls*) [source]

Sets the number of channels of the meter.

Args: nchnls: int

The number of channels.

PyoGuiGrapher

class **PyoGuiGrapher**(*parent, xlen=8192, yrange=(0, 1), init=[(0.0, 0.0), (1.0, 1.0)], mode=0, exp=10, inverse=True, tension=0, bias=0, pos=(0, 0), size=(300, 200), style=0*) [source]

Multi-modes break-points function editor.

Parent: wx.Panel

Events: EVT_PYO_GUI_GRAPHER

Sent after any change of the grapher function. The current list of points of the grapher can be retrieve with the *value* attribute of the generated event. The object itself can be retrieve with the *object* attribute of the event and the object's id with the *id* attrbute.

Args: parent: wx.Window

The parent window.

xlen: int, optional

The length, in samples, of the grapher. Defaults to 8192.

yrange: two-values tuple, optional

A tuple indicating the minimum and maximum values of the Y-axis. Defaults to (0, 1).

init: list of two-values tuples, optional

The initial break-points function set as normalized values. A point is defined with its X and Y positions as a tuple. Defaults to [(0.0, 0.0), (1.0, 1.0)].

mode: int, optional

The grapher mode definning how line segments will be draw. Possible modes are:

0. linear (default)
1. cosine
2. exponential (uses *exp* and *inverse* arguments)
3. curve (uses *tension* and *bias* arguments)
4. logarithmic
5. logarithmic cosine

exp: int or float, optional

The exponent factor for an exponential graph. Defaults to 10.0.

inverse: boolean, optional

If True, downward slope will be inversed. Useful to create biexponential curves. Defaults to True.

tension: int or float, optional

Curvature at the known points. 1 is high, 0 normal, -1 is low. Defaults to 0.

bias: int or float, optional

Curve attraction (for each segments) toward bundary points. 0 is even, positive is towards first point, negative is towards the second point. Defaults to 0.

pos: wx.Point, optional

Window position in pixels. Defaults to (0, 0).

size: wx.Size, optional

Window size in pixels. Defaults to (300, 200).

style: int, optional

Window style (see wx.Window documentation). Defaults to 0.

`reset`() [source]

Resets the points to the initial state.

`getPoints`() [source]

Returns the current normalized points of the grapher.

`getValues`() [source]

Returns the current points, according to Y-axis range, of the grapher.

`setPoints`(*pts*) [source]

Sets a new group of normalized points in the grapher.

Args: pts: list of two-values tuples

New normalized (between 0 and 1) points.

`setValues`(*vals*) [source]

Sets a new group of points, according to Y-axis range, in the grapher.

Args: vals: list of two-values tuples

New real points.

`setYrange`(*yrange*) [source]

Sets a new Y-axis range to the grapher.

Args: yrange: two-values tuple

New Y-axis range.

`setInitPoints`(*pts*) [source]

Sets a new initial normalized points list to the grapher.

Args: pts: list of two-values tuples

New normalized (between 0 and 1) initial points.

`setMode`(*x*) [source]

Changes the grapher's mode.

Args: x: int

New mode. Possible modes are:

0. linear (default)
1. cosine
2. exponential (uses *exp* and *inverse* arguments)
3. curve (uses *tension* and *bias* arguments)
4. logarithmic
5. logarithmic cosine

`setExp`(*x*) [source]

Changes the grapher's exponent factor for exponential graph.

Args: x: float

New exponent factor.

`setInverse`(*x*) [source]

Changes the grapher's inverse boolean for exponential graph.

Args: x: boolean

New inverse factor.

`setTension`(*x*) [source]

Changes the grapher's tension factor for curved graph.

Args: x: float

New tension factor.

`setBias`(*x*) [source]

Changes the grapher's bias factor for curved graph.

Args: x: float

New bias factor.

PyoGuiMultiSlider

class `PyoGuiMultiSlider`(*parent*, *xlen=16*, *yrange=(0, 1)*, *init=None*, *pos=(0, 0)*, *size=(300, 200)*, *style=0*)

Data multi-sliders editor. [source]

Parent: wx.Panel

Events: EVT_PYO_GUI_MULTI_SLIDER

Sent after any change of the multi-sliders values. The current list of values of the multi-sliders can be retrieve with the *value* attribute of the generated event. The object itself can be retrieve with the *object* attribute of the event and the object's id with the *id* attrbute.

Args: parent: wx.Window

The parent window.

xlen: int, optional

The number of sliders in the multi-sliders. Defaults to 16.

yrange: two-values tuple

A tuple indicating the minimum and maximum values of the Y-axis. Defaults to (0, 1).

init: list values, optional

The initial list of values of the multi-sliders. Defaults to None, meaning all sliders initialized to the minimum value.

pos: wx.Point, optional

Window position in pixels. Defaults to (0, 0).

size: wx.Size, optional

Window size in pixels. Defaults to (300, 200).

style: int, optional

Window style (see wx.Window documentation). Defaults to 0.

`reset`() [source]

Resets the sliders to their initial state.

`getValues`() [source]

Returns the current values of the sliders.

`setValues`(*vals*) [source]

Sets new values to the sliders.

Args: vals: list of values

New values.

`setYrange`(*yrange*) [source]

Sets a new Y-axis range to the multi-sliders.

Args: yrange: two-values tuple

New Y-axis range.

PyoGuiSpectrum

class **PyoGuiSpectrum**(*parent*, *lowfreq=0*, *highfreq=22050*, *fscaling=0*, *mscaling=0*, *pos=(0, 0)*, *size=(300, 200)*, *style=0*) [source]

Frequency spectrum display.

This widget should be used with the Spectrum object, which measures the magnitude of an input signal versus frequency within a user defined range. It can show both magnitude and frequency on linear or logarithmic scale.

To create the bridge between the analyzer and the display, the Spectrum object must be registered in the PyoGuiSpectrum object with the setAnalyzer(obj) method. The Spectrum object will automatically call the update(points) method to refresh the display.

Parent: wx.Panel

Args: parent: wx.Window

The parent window.

lowfreq: int or float, optional

The lowest frequency, in Hz, to display on the X-axis. Defaults to 0.

highfreq: int or float, optional

The highest frequency, in Hz, to display on the X-axis. Defaults to 22050.

fscaling: int, optional

The frequency scaling on the X-axis. 0 means linear, 1 means logarithmic. Defaults to 0.

mscaling: int, optional

The magnitude scaling on the Y-axis. 0 means linear, 1 means logarithmic. Defaults to 0.

pos: wx.Point, optional

Window position in pixels. Defaults to (0, 0).

size: wx.Size, optional

Window size in pixels. Defaults to (300, 200).

style: int, optional

Window style (see wx.Window documentation). Defaults to 0.

update(*points*) [source]

Display updating method.

This method is automatically called by the audio analyzer object (Spectrum) with points to draw as arguments. The points are already formatted for the current drawing surface to save CPU cycles.

The method setAnalyzer(obj) must be used to register the audio analyzer object.

Args: points: list of list of tuples

A list containing n-channels list of tuples. A tuple is a point (X-Y coordinates) to draw.

`setAnalyzer`(*object*) [source]

Register an audio analyzer object (Spectrum).

Args: object: Spectrum object

The audio object performing the frequency analysis.

`setLowFreq`(*x*) [source]

Changes the lowest frequency of the display.

This method propagates the value to the audio analyzer.

Args: x: int or float

New lowest frequency.

`setHighFreq`(*x*) [source]

Changes the highest frequency of the display.

This method propagates the value to the audio analyzer.

Args: x: int or float

New highest frequency.

`setFscaling`(*x*) [source]

Changes the frequency scaling (X-axis) of the display.

This method propagates the value to the audio analyzer.

Args: x: int

0 means linear scaling, 1 means logarithmic scaling.

`setMscaling`(*x*) [source]

Changes the magnitude scaling (Y-axis) of the display.

This method propagates the value to the audio analyzer.

Args: x: int

0 means linear scaling, 1 means logarithmic scaling.

PyoGuiScope

class **PyoGuiScope**(*parent*, *length=0.05*, *gain=0.67*, *pos=(0, 0)*, *size=(300, 200)*, *style=0*) [source]

Oscilloscope display.

This widget should be used with the Scope object, which computes the waveform of an input signal to display on a GUI.

To create the bridge between the analyzer and the display, the Scope object must be registered in the PyoGuiScope object with the setAnalyzer(obj) method. The Scope object will automatically call the update(points) method to refresh the display.

Parent: wx.Panel

Args: parent: wx.Window

The parent window.

length: float, optional

Length, in seconds, of the waveform segment displayed on the window. Defaults to 0.05.

gain: float, optional

Linear gain applied to the signal to be displayed. Defaults to 0.67.

pos: wx.Point, optional

Window position in pixels. Defaults to (0, 0).

size: wx.Size, optional

Window size in pixels. Defaults to (300, 200).

style: int, optional

Window style (see wx.Window documentation). Defaults to 0.

update(*points*) [source]

Display updating method.

This method is automatically called by the audio analyzer object (Scope) with points to draw as arguments. The points are already formatted for the current drawing surface to save CPU cycles.

The method setAnalyzer(obj) must be used to register the audio analyzer object.

Args: points: list of list of tuples

A list containing n-channels list of tuples. A tuple is a point (X-Y coordinates) to draw.

setAnalyzer(*object*) [source]

Register an audio analyzer object (Scope).

Args: object: Scope object

The audio object performing the waveform analysis.

setLength(*x*) [source]

Changes the length, in seconds, of the displayed audio segment.

This method propagates the value to the audio analyzer.

Args: x: float

New segment length in seconds.

setGain(*x*) [source]

Changes the gain applied to the input signal.

This method propagates the value to the audio analyzer.

Args: x: float

New linear gain.

PyoGuiSndView

class **PyoGuiSndView**(*parent*, *pos=(0, 0)*, *size=(300, 200)*, *style=0*) [source]

Soundfile display.

This widget should be used with the SndTable object, which keeps soundfile in memory and computes the waveform to display on the GUI.

To create the bridge between the audio memory and the display, the SndTable object must be registered in the PyoGuiSndView object with the setTable(object) method.

The SndTable object will automatically call the update() method to refresh the display when the table is modified.

Parent: wx.Panel

Events: EVT_PYO_GUI_SNDVIEW_MOUSE_POSITION

Sent when the mouse is moving on the panel with the left button pressed. The *value* attribute of the event will hold the normalized position of the mouse into the sound. For X-axis value, 0.0 is the beginning of the sound and 1.0 is the end of the sound. For the Y-axis, 0.0 is the bottom of the panel and 1.0 is the top. The object itself can be retrieve with the *object* attribute of the event and the object's id with the *id* attrbute.

EVT_PYO_GUI_SNDVIEW_SELECTION

Sent when a new region is selected on the panel. A new selection is created with a Right-click and drag on the panel. The current selection can be moved with Shift+Right-click and drag. Ctrl+Right-click (Cmd on OSX) remove the selected

region. The *value* attribute of the event will hold the normalized selection as a tuple (min, max). 0.0 means the beginning of the sound and 1.0 means the end of the sound. The object itself can be retrieve with the *object* attribute of the event and the object's id with the *id* attrbute.

Args: parent: wx.Window

The parent window.

pos: wx.Point, optional

Window position in pixels. Defaults to (0, 0).

size: wx.Size, optional

Window size in pixels. Defaults to (300, 200).

style: int, optional

Window style (see wx.Window documentation). Defaults to 0.

`update`() [source]

Display updating method.

This method is automatically called by the audio memory object (SndTable) when the table is modified.

The method setTable(obj) must be used to register the audio memory object.

`setTable`(*object*) [source]

Register an audio memory object (SndTable).

Args: object: SndTable object

The audio table keeping the sound in memory.

`setSelection`(*start*, *stop*) [source]

Changes the selected region.

This method will trigger a EVT_PYO_GUI_SNDVIEW_SELECTION event with a tuple (start, stop) as value.

Args: start: float

The starting point of the selected region. This value must be normalized between 0 and 1 (0 is the beginning of the sound, 1 is the end).

stop: float

The ending point of the selected region. This value must be normalized between 0 and 1 (0 is the beginning of the sound, 1 is the end).

`resetSelection`() [source]

Removes the selected region.

This method will trigger a EVT_PYO_GUI_SNDVIEW_SELECTION event with a tuple (0.0, 1.0) as value.