xSGE GUI Toolkit Documentation

Release 1.1

onpon4

June 09, 2017

CONTENTS

Contents

xSGE is a collection of extensions for the SGE licensed under the GNU General Public License. They are designed to give additional features to free/libre software games which aren't necessary, but are nice to have.

xSGE extensions are not dependent on any particular SGE implementation. They should work with any implementation that follows the specification.

This extension provides a simple toolkit for adding GUIs to a SGE game as well as support for modal dialog boxes.

To use this extension, you must call *xsge_gui.init()* sometime between the creation of the `sge.dsp.Game` object and the start of the game.

`xsge_gui.`**`window_background_color`**
`xsge_gui.`**`keyboard_focused_box_color`**
`xsge_gui.`**`text_color`**
`xsge_gui.`**`button_text_color`**
`xsge_gui.`**`textbox_text_color`**
`xsge_gui.`**`textbox_text_selected_color`**
`xsge_gui.`**`textbox_highlight_color`**
`xsge_gui.`**`title_text_color`**

The colors used by this module. They can be safely changed, but be sure to call `redraw()` on all windows and widgets that would be affected; some changes might not become visible until you do.

xsge_gui.**default_font**
xsge_gui.**button_font**
xsge_gui.**textbox_font**
xsge_gui.**title_font**
The fonts used by this module. They can be safely changed, but be sure to call `redraw()` on all windows and widgets that would be affected; some changes might not become visible until you do.

xsge_gui.**button_sprite**
xsge_gui.**button_left_sprite**
xsge_gui.**button_right_sprite**
xsge_gui.**button_pressed_sprite**
xsge_gui.**button_pressed_left_sprite**
xsge_gui.**button_pressed_right_sprite**
xsge_gui.**button_selected_sprite**
xsge_gui.**button_selected_left_sprite**
xsge_gui.**button_selected_right_sprite**
xsge_gui.**checkbox_off_sprite**
xsge_gui.**checkbox_on_sprite**
xsge_gui.**progressbar_sprite**
xsge_gui.**progressbar_left_sprite**
xsge_gui.**progressbar_right_sprite**
xsge_gui.**progressbar_container_sprite**
xsge_gui.**progressbar_container_left_sprite**
xsge_gui.**progressbar_container_right_sprite**
xsge_gui.**radiobutton_off_sprite**
xsge_gui.**radiobutton_on_sprite**
xsge_gui.**textbox_sprite**
xsge_gui.**textbox_left_sprite**
xsge_gui.**textbox_right_sprite**
xsge_gui.**window_border_left_sprite**
xsge_gui.**window_border_right_sprite**
xsge_gui.**window_border_bottom_sprite**
xsge_gui.**window_border_bottomleft_sprite**
xsge_gui.**window_border_bottomright_sprite**
xsge_gui.**window_border_top_sprite**
xsge_gui.**window_border_topleft_sprite**
xsge_gui.**window_border_topright_sprite**
The sprites used by this module. They can be safely changed, but be sure to call `redraw()` on all windows and widgets that would be affected; some changes might not become visible until you do.

xsge_gui.**next_window_keys**
A list of keys which, when pressed, will give keyboard focus to the next available window.

Default value: `[]`

xsge_gui.**previous_window_keys**
A list of keys which, when pressed, will give keyboard focus to the previous available window.

Default value: `[]`

xsge_gui.**next_widget_keys**
A list of keys which, when pressed, will give keyboard focus to the next available widget on the window which has keyboard focus.

Default value: `["tab"]`

xsge_gui.**previous_widget_keys**
A list of keys which, when pressed, will give keyboard focus to the previous available widget on the window which has keyboard focus.

Default value: `[]`

xsge_gui.**left_keys**
A list of keys to treat as left arrows.

Default value: `["left"]`

xsge_gui.**right_keys**
A list of keys to treat as right arrows.

Default value: `["right"]`

xsge_gui.**up_keys**
A list of keys to treat as up arrows.

Default value: `["up"]`

xsge_gui.**down_keys**
A list of keys to treat as down arrows.

Default value: `["down"]`

xsge_gui.**enter_keys**
A list of keys to treat as Enter keys.

Default value: `["enter", "kp_enter"]`

xsge_gui.**escape_keys**
A list of keys to treat as Escape keys.

Default value: `["escape"]`

xsge_gui.**next_window_joystick_events**
A list of tuples indicating joystick events which will give keyboard focus to the next available window.

The tuples should contain, in order, the following values:

1. The number of the joystick, where `0` is the first joystick.
2. The type of event. See the documentation for `sge.input.JoystickEvent` for more information.
3. The number of the joystick control, where `0` is the first control of its type on the joystick.

Default value: `[]`

xsge_gui.**previous_window_joystick_events**
A list of tuples indicating joystick events which will give keyboard focus to the previous available window.

See the documentation for *next_window_joystick_events* for more information.

Default value: `[]`

xsge_gui.**next_widget_joystick_events**
A list of tuples indicating joystick events which will give keyboard focus to the next available widget on the window which has keyboard focus.

See the documentation for *next_window_joystick_events* for more information.

Default value: `[(0, "button", 8)]`

xsge_gui.**previous_widget_joystick_events**
A list of tuples indicating joystick events which will give keyboard focus to the previous available widget on the window which has keyboard focus.

See the documentation for *next_window_joystick_events* for more information.

Default value: `[]`

xsge_gui.**left_joystick_events**
A list of tuples indicating joystick events to treat as left arrows.

See the documentation for *next_window_joystick_events* for more information.

Default value: `[(0, "axis-", 0), (0, "hat_left", 0)]`

xsge_gui.**right_joystick_events**
A list of tuples indicating joystick events to treat as right arrows.

See the documentation for *next_window_joystick_events* for more information.

Default value: `[(0, "axis+", 0), (0, "hat_right", 0)]`

xsge_gui.**up_joystick_events**
A list of tuples indicating joystick events to treat as up arrows.

See the documentation for *next_window_joystick_events* for more information.

Default value: `[(0, "axis-", 1), (0, "hat_up", 0)]`

xsge_gui.**down_joystick_events**
A list of tuples indicating joystick events to treat as down arrows.

See the documentation for *next_window_joystick_events* for more information.

Default value: `[(0, "axis+", 1), (0, "hat_down", 0)]`

xsge_gui.**enter_joystick_events**
A list of tuples indicating joystick events to treat as Enter keys.

See the documentation for *next_window_joystick_events* for more information.

Default value: `[(0, "button", 9)]`

xsge_gui.**escape_joystick_events**
A list of tuples indicating joystick events to treat as Escape keys.

Default value: `[]`

xsge_gui.**joystick_threshold**
The amount of tilt on a joystick that should be considered "triggered" for the purpose of navigating menus.

CHAPTER

ONE

XSGE_GUI CLASSES

xsge_gui.Handler

class xsge_gui.**Handler**

An object of this class needs to exist in any room where windows are to be used. It feeds SGE events to the windows so they can react to user input. It also refreshes all windows every frame.

windows

A list of all windows that are currently handled by this handler.

You don't need to modify this list manually. Instead, use *xsge_gui.Window.show()* and *xsge_gui.Window.hide()* to add and remove windows from this list, respectively.

keyboard_focused_window

The window that currently has keyboard focus, or None if no window has focus.

xsge_gui.Handler Methods

Handler.**get_mouse_focused_window**()

Return the window that currently has mouse focus. The window with mouse focus is the one which is closest to the front that is touching the mouse cursor.

Return None if no window has focus.

xsge_gui.Window

class xsge_gui.**Window**(*parent*, *x*, *y*, *width*, *height*, *title=''*, *background_color=None*, *border=True*)

Window objects are used to contain widgets. They can be moved around the game window by the user.

parent

A weak reference to this window's parent handler object, which is used to display it when it is supposed to be visible.

If a strong reference is assigned to this attribute, it will automatically be changed to a weak reference.

x

The horizontal position of the window relative to the game window.

y

The vertical position of the window relative to the game window.

width

The width of the window.

height
The height of the window.

title
The text that shows up in the title bar of the window.

background_color
The color of this window's background. If set to `None`, it becomes the same value as *xsge_gui.window_background_color*.

border
Whether or not the window has a border. If this is `False`, the window cannot be moved or resized by the user, and *title* will not be displayed.

widgets
A list of this window's widgets.

keyboard_focused_widget
The widget which currently has keyboard focus within this window, or `None` if no widget has keyboard focus within this window.

sprite
The sprite this window currently displays as itself.

xsge_gui.Window Methods

`Window.`**`show`**`()`
Add this window to its parent handler.

`Window.`**`hide`**`()`
Remove this window from its parent handler.

`Window.`**`move_to_front`**`()`
Move this window in front of all other windows.

`Window.`**`move_to_back`**`()`
Move this window behind all other windows.

`Window.`**`destroy`**`()`
An alias for *xsge_gui.Window.hide()*.

`Window.`**`redraw`**`()`
Re-draw this window's sprite.

Call this method if you change any variables that should affect this window's appearance. For performance reasons, the changes won't show up in an existing window until this method is called.

`Window.`**`refresh`**`()`
Project this window onto the game window.

This method must be called every frame for the window to be visible.

`Window.`**`get_mouse_on_titlebar`**`()`
Return whether or not the mouse is on the title bar.

`Window.`**`get_mouse_focused_widget`**`()`
Return the widget in this window with mouse focus. The widget with mouse focus is the one which is closest to the front that is touching the mouse cursor.

Return `None` if no widget has focus.

xsge_gui.Window Event Methods

`Window.`**`event_step`**(*time_passed*, *delta_mult*)
Called once every frame, before refreshing. See the documentation for `sge.dsp.Game.event_step()` for more information.

`Window.`**`event_change_keyboard_focus`**()
Called when *keyboard_focused_widget* changes as a result of a key or joystick event being pressed.

`Window.`**`event_key_press`**(*key*, *char*)
Called when a key is pressed while this window has keyboard focus. See the documentation for `sge.input.KeyPress` for more information.

`Window.`**`event_key_release`**(*key*)
Called when a key is released while this window has keyboard focus. See the documentation for `sge.input.KeyRelease` for more information.

`Window.`**`event_mouse_button_press`**(*button*)
Called when a mouse button is pressed while this window has mouse focus. See the documentation for `sge.input.MouseButtonPress` for more information.

`Window.`**`event_mouse_button_release`**(*button*)
Called when a mouse button is released while this window has mouse focus. See the documentation for `sge.input.MouseButtonRelease` for more information.

`Window.`**`event_joystick_axis_move`**(*js_name*, *js_id*, *axis*, *value*)
Called when a joystick axis is moved while this window has keyboard focus. See the documentation for `sge.input.JoystickAxisMove` for more information.

`Window.`**`event_joystick_hat_move`**(*js_name*, *js_id*, *hat*, *x*, *y*)
Called when a joystick hat is moved while this window has keyboard focus. See the documentation for `sge.input.JoystickHatMove` for more information.

`Window.`**`event_joystick_trackball_move`**(*js_name*, *js_id*, *ball*, *x*, *y*)
Called when a joystick trackball is moved while this window has keyboard focus. See the documentation for `sge.input.JoystickTrackballMove` for more information.

`Window.`**`event_joystick_button_press`**(*js_name*, *js_id*, *button*)
Called when a joystick button is pressed while this window has keyboard focus. See the documentation for `sge.input.JoystickButtonPress` for more information.

`Window.`**`event_joystick_button_release`**(*js_name*, *js_id*, *button*)
Called when a joystick button is released while this window has keyboard focus. See the documentation for `sge.input.JoystickButtonRelease` for more information.

`Window.`**`event_joystick`**(*js_name*, *js_id*, *input_type*, *input_id*, *value*)
Called when a joystick event occurs while this window has keyboard focus. See the documentation for `sge.inputJoystickEvent` for more information.

`Window.`**`event_press_left`**()
Called when a key in *left_keys* or a joystick event in *left_joystick_events* is pressed while this window has keyboard focus.

`Window.`**`event_press_right`**()
Called when a key in *right_keys* or a joystick event in *right_joystick_events* is pressed while this window has keyboard focus.

`Window.`**`event_press_up`**()
Called when a key in *up_keys* or a joystick event in *up_joystick_events* is pressed while this window has keyboard focus.

Window.**event_press_down**()
Called when a key in *down_keys* or a joystick event in *down_joystick_events* is pressed while this window has keyboard focus.

Window.**event_press_enter**()
Called when a key in *enter_keys* or a joystick event in *enter_joystick_events* is pressed while this window has keyboard focus.

Window.**event_press_escape**()
Called when a key in *escape_keys* or a joystick event in *enter_joystick_events* is pressed while this window has keyboard focus.

Window.**event_release_left**()
Called when a key in *left_keys* or a joystick event in *left_joystick_events* is released while this window has keyboard focus.

Window.**event_release_right**()
Called when a key in *right_keys* or a joystick event in *right_joystick_events* is released while this window has keyboard focus.

Window.**event_release_up**()
Called when a key in *up_keys* or a joystick event in *up_joystick_events* is released while this window has keyboard focus.

Window.**event_release_down**()
Called when a key in *down_keys* or a joystick event in *down_joystick_events* is released while this window has keyboard focus.

Window.**event_release_enter**()
Called when a key in *enter_keys* or a joystick event in *enter_joystick_events* is released while this window has keyboard focus.

Window.**event_release_escape**()
Called when a key in *escape_keys* or a joystick event in *enter_joystick_events* is released while this window has keyboard focus.

Window.**event_titlebar_mouse_button_press**(*button*)
Called when a mouse button is pressed on top of this window's title bar (top border). See the documentation for `sge.input.MouseButtonPress` for more information.

Window.**event_titlebar_mouse_button_release**(*button*)
Called when a mouse button is released on top of this window's title bar (top border). See the documentation for `sge.input.MouseButtonRelease` for more information.

Window.**event_global_key_press**(*key*, *char*)
Called when a key is pressed, regardless of which window has keyboard focus. See the documentation for `sge.input.KeyPress` for more information.

Window.**event_global_key_release**(*key*)
Called when a key is released, regardless of which window has keyboard focus. See the documentation for `sge.input.KeyRelease` for more information.

Window.**event_global_mouse_button_press**(*button*)
Called when a mouse button is pressed, regardless of which window has mouse focus. See the documentation for `sge.input.MouseButtonPress` for more information.

Window.**event_global_mouse_button_release**(*button*)
Called when a mouse button is released, regardless of which window has mouse focus. See the documentation for `sge.input.MouseButtonRelease` for more information.

`Window.`**`event_global_joystick_axis_move`**(*js_name*, *js_id*, *axis*, *value*)
Called when a joystick axis is moved, regardless of which window has keyboard focus. See the documentation for `sge.input.JoystickAxisMove` for more information.

`Window.`**`event_global_joystick_hat_move`**(*js_name*, *js_id*, *hat*, *x*, *y*)
Called when a joystick hat is moved, regardless of which window has keyboard focus. See the documentation for `sge.input.JoystickHatMove` for more information.

`Window.`**`event_global_joystick_trackball_move`**(*js_name*, *js_id*, *ball*, *x*, *y*)
Called when a joystick trackball is moved, regardless of which window has keyboard focus. See the documentation for `sge.input.JoystickTrackballMove` for more information.

`Window.`**`event_global_joystick_button_press`**(*js_name*, *js_id*, *button*)
Called when a joystick button is pressed, regardless of which window has keyboard focus. See the documentation for `sge.input.JoystickButtonPress` for more information.

`Window.`**`event_global_joystick_button_release`**(*js_name*, *js_id*, *button*)
Called when a joystick button is released, regardless of which window has keyboard focus. See the documentation for `sge.input.JoystickButtonRelease` for more information.

`Window.`**`event_global_joystick`**(*js_name*, *js_id*, *input_type*, *input_id*, *value*)
Called when a joystick event occurs, regardless of which window has keyboard focus. See the documentation for `sge.inputJoystickEvent` for more information.

`Window.`**`event_close`**()
Called when the "X" button in the top-right corner of the window is pressed.

By default, this calls *xsge_gui.Window.destroy()*.

xsge_gui.Dialog

class `xsge_gui.`**`Dialog`**(*parent*, *x*, *y*, *width*, *height*, *title=''*, *background_color=None*, *border=True*)
Dialog class.

Dialogs are windows with their own loops, also called modal windows. They are used for tasks that must be completed before the main program continues, such as pop-up messages.

See the documentation for *xsge_gui.Window* for more information.

xsge_gui.Dialog Methods

In addition to methods inherited from *xsge_gui.Window*, the following methods are also available:

`Dialog.`**`show`**()
Show this dialog and start its loop.

Like *xsge_gui.Window.show()*, this method adds the dialog to its parent. It then starts this dialog's loop. Call `xsge_gui.Dialog.hide()` on this dialog to end the loop.

xsge_gui.MenuWindow

class `xsge_gui.`**`MenuWindow`**(*parent*, *x*, *y*, *width*, *height*, *title=''*, *background_color=sge.gfx.Color("#00000000")*, *border=False*)
Meant to be used with *xsge_gui.MenuItem* widgets to create keyboard-navigated menus. Has no border by default. If one of the keys in *enter_keys* or one of the joystick events in *enter_joystick_events*

is pressed, *choice* is set to the index within `widgets` of the widget which currently has keyboard focus, the window is closed, and *event_choose()* is called. If one of the keys in *escape_keys* or one of the joystick events in *escape_joystick_events* is pressed, the window is closed and *event_choose()* is called.

choice
The menu item chosen. If no menu item has been chosen, it is set to `None`.

See the documentation for *xsge_gui.Window* for more information.

xsge_gui.MenuWindow Methods

In addition to the methods inherited from *xsge_gui.Window*, the following methods are also available:

classmethod `MenuWindow.`**`from_text`**(*parent*, *x*, *y*, *items*, *font_normal=None*, *color_normal=None*, *font_selected=None*, *color_selected=None*, *background_color=sge.gfx.Color("#00000000")*, *height=None*, *margin=0*, *halign='left'*, *valign='top'*)
Return a menu created automatically from a list of strings.

Arguments:

- `x` – The horizontal location of the window within the room. Affected by `halign`.
- `y` – The vertical location of the window within the room. Affected by `valign`.
- `items` – A list of strings to use as the menu's items.
- `font_normal` – The default font to use.
- `color_normal` – The default color to use.
- `font_selected` – The font to use for the currently selected item. If set to `None`, the default font will be used.
- `color_selected` – The color to use for the currently selected item. If set to `None`, the default color will be used.
- `height` – The height of the menu. If set to `None`, it will be the sum of the items' height.
- `margin` – The size of the margin around the menu.
- `halign` – The horizontal alignment of the menu. See the documentation for `sge.gfx.Sprite.draw_text()` for more information.
- `valign` – The vertical alignment of the menu. See the documentation for `sge.gfx.Sprite.draw_text()` for more information.

xsge_gui.MenuWindow Event Methods

In addition to the event methods inherited from *xsge_gui.Window*, the following event methods are also available:

`MenuWindow.`**`event_choose`**`()`
Called when a menu item is chosen.

xsge_gui.MenuDialog

class xsge_gui.**MenuDialog**(*parent*, *x*, *y*, *width*, *height*, *title=''*, *background_color=sge.gfx.Color("#00000000")*, *border=False*)

Inherits both *MenuWindow* and *Dialog*.

See the documentation for *xsge_gui.Dialog* for more information.

xsge_gui.MessageDialog

class xsge_gui.**MessageDialog**(*parent*, *message=''*, *title='Message'*, *buttons=('Ok',)*, *default=-1*, *width=320*, *height=None*)

This dialog shows a message box and accepts button input. All buttons cause the dialog to close and set *choice* to the button pressed.

choice

The button clicked. If a button hasn't been clicked (i.e. the dialog hasn't yet been closed or was closed by clicking on the close button), it is set to None.

See the documentation for *xsge_gui.Dialog* for more information.

xsge_gui.TextEntryDialog

class xsge_gui.**TextEntryDialog**(*parent*, *message=''*, *title='Text Entry'*, *text=''*, *width=320*, *height=None*)

This dialog shows a message and has the user enter some text. Two buttons are shown: a "Cancel" button that closes the dialog, and an "Ok" button that sets *text* to the text entered and then closes the dialog.

text

The text entered after the "Ok" button is clicked. If the "Ok" button hasn't been clicked, this is None.

See the documentation for *xsge_gui.Dialog* for more information.

xsge_gui.Widget

class xsge_gui.**Widget**(*parent*, *x*, *y*, *z*, *sprite=None*, *index=None*)

Widget objects are things like controls and decorations that exist on windows.

parent

A weak reference to this widget's parent window.

If a strong reference is assigned to this attribute, it will automatically be changed to a weak reference.

x

The horizontal position of the widget relative to its parent window.

y

The vertical position of the widget relative to its parent window.

z

The Z-axis position of the widget. Widgets with a higher Z-axis value are in front of widgets with a lower Z-axis value. This value is not connected in any way to Z-axis values in the SGE.

sprite
The sprite this widget displays as itself.

index
Indicates the "position" this widget is in for the purposes of tab-focusing. Smaller indexes are first on the window's list of widgets. Set to `None` to use `z` for this purpose. Widgets with smaller indexes are inserted earlier in the parent window's `widgets` list, and this positioning is what actually determines the ordering of tab-focusing.

Note: This attribute does not *precisely* control the index of the widget within the parent widget's `widgets` list. It only controls where the widget is in the list relative to other widgets.

---

tab_focus
Class attribute indicating whether or not the widget should be considered for focusing when the Tab key is pressed.

Default value: `True`

xsge_gui.Widget Methods

`Widget.`**`destroy`**`()`
Destroy this widget.

`Widget.`**`redraw`**`()`
Re-draw this widget's sprite.

Call this method if you change any variables that should affect this widget's appearance. This method automatically makes any changes necessary to `self.sprite`.

`Widget.`**`refresh`**`()`
Project this widget onto the game window.

This method must be called every frame for the widget to be visible.

xsge_gui.Widget Event Methods

`Widget.`**`event_step`**`(`*time_passed*, *delta_mult*`)`
Called once every frame, before refreshing. See the documentation for `sge.dsp.Game.event_step()` for more information.

`Widget.`**`event_key_press`**`(`*key*, *char*`)`
Called when a key is pressed while this widget has keyboard focus. See the documentation for `sge.input.KeyPress` for more information.

`Widget.`**`event_key_release`**`(`*key*`)`
Called when a key is released while this widget has keyboard focus. See the documentation for `sge.input.KeyRelease` for more information.

`Widget.`**`event_mouse_button_press`**`(`*button*`)`
Called when a mouse button is pressed while this widget has mouse focus. See the documentation for `sge.input.MouseButtonPress` for more information.

`Widget.`**`event_mouse_button_release`**`(`*button*`)`
Called when a mouse button is released while this widget has mouse focus. See the documentation for `sge.input.MouseButtonRelease` for more information.

Widget.**event_joystick_axis_move**(*js_name*, *js_id*, *axis*, *value*)
Called when a joystick axis is moved while this widget has keyboard focus. See the documentation for `sge.input.JoystickAxisMove` for more information.

Widget.**event_joystick_hat_move**(*js_name*, *js_id*, *hat*, *x*, *y*)
Called when a joystick hat is moved while this widget has keyboard focus. See the documentation for `sge.input.JoystickHatMove` for more information.

Widget.**event_joystick_trackball_move**(*js_name*, *js_id*, *ball*, *x*, *y*)
Called when a joystick trackball is moved while this widget has keyboard focus. See the documentation for `sge.input.JoystickTrackballMove` for more information.

Widget.**event_joystick_button_press**(*js_name*, *js_id*, *button*)
Called when a joystick button is pressed while this widget has keyboard focus. See the documentation for `sge.input.JoystickButtonPress` for more information.

Widget.**event_joystick_button_release**(*js_name*, *js_id*, *button*)
Called when a joystick button is released while this widget has keyboard focus. See the documentation for `sge.input.JoystickButtonRelease` for more information.

Widget.**event_joystick**(*js_name*, *js_id*, *input_type*, *input_id*, *value*)
Called when a joystick event occurs while this widget has keyboard focus. See the documentation for `sge.input.JoystickEvent` for more information.

Widget.**event_press_left**()
Called when a key in *left_keys* or a joystick event in *left_joystick_events* is pressed while this widget has keyboard focus.

Widget.**event_press_right**()
Called when a key in *right_keys* or a joystick event in *right_joystick_events* is pressed while this widget has keyboard focus.

Widget.**event_press_up**()
Called when a key in *up_keys* or a joystick event in *up_joystick_events* is pressed while this widget has keyboard focus.

Widget.**event_press_down**()
Called when a key in *down_keys* or a joystick event in *down_joystick_events* is pressed while this widget has keyboard focus.

Widget.**event_press_enter**()
Called when a key in *enter_keys* or a joystick event in *enter_joystick_events* is pressed while this widget has keyboard focus.

Widget.**event_press_escape**()
Called when a key in *escape_keys* or a joystick event in *enter_joystick_events* is pressed while this widget has keyboard focus.

Widget.**event_release_left**()
Called when a key in *left_keys* or a joystick event in *left_joystick_events* is released while this widget has keyboard focus.

Widget.**event_release_right**()
Called when a key in *right_keys* or a joystick event in *right_joystick_events* is released while this widget has keyboard focus.

Widget.**event_release_up**()
Called when a key in *up_keys* or a joystick event in *up_joystick_events* is released while this widget has keyboard focus.

Widget.**event_release_down**()
Called when a key in *down_keys* or a joystick event in *down_joystick_events* is released while this widget has keyboard focus.

Widget.**event_release_enter**()
Called when a key in *enter_keys* or a joystick event in *enter_joystick_events* is released while this widget has keyboard focus.

Widget.**event_release_escape**()
Called when a key in *escape_keys* or a joystick event in *enter_joystick_events* is released while this widget has keyboard focus.

Widget.**event_global_key_press**(*key*, *char*)
Called when a key is pressed, regardless of which widget has keyboard focus. See the documentation for `sge.input.KeyPress` for more information.

Widget.**event_global_key_release**(*key*)
Called when a key is released, regardless of which widget has keyboard focus. See the documentation for `sge.input.KeyRelease` for more information.

Widget.**event_global_mouse_button_press**(*button*)
Called when a mouse button is pressed, regardless of which widget has mouse focus. See the documentation for `sge.input.MouseButtonPress` for more information.

Widget.**event_global_mouse_button_release**(*button*)
Called when a mouse button is released, regardless of which widget has mouse focus. See the documentation for `sge.input.MouseButtonRelease` for more information.

Widget.**event_global_joystick_axis_move**(*js_name*, *js_id*, *axis*, *value*)
Called when a joystick axis is moved, regardless of which widget has keyboard focus. See the documentation for `sge.input.JoystickAxisMove` for more information.

Widget.**event_global_joystick_hat_move**(*js_name*, *js_id*, *hat*, *x*, *y*)
Called when a joystick hat is moved, regardless of which widget has keyboard focus. See the documentation for `sge.input.JoystickHatMove` for more information.

Widget.**event_global_joystick_trackball_move**(*js_name*, *js_id*, *ball*, *x*, *y*)
Called when a joystick trackball is moved, regardless of which widget has keyboard focus. See the documentation for `sge.input.JoystickTrackballMove` for more information.

Widget.**event_global_joystick_button_press**(*js_name*, *js_id*, *button*)
Called when a joystick button is pressed, regardless of which widget has keyboard focus. See the documentation for `sge.input.JoystickButtonPress` for more information.

Widget.**event_global_joystick_button_release**(*js_name*, *js_id*, *button*)
Called when a joystick button is released, regardless of which widget has keyboard focus. See the documentation for `sge.input.JoystickButtonRelease` for more information.

xsge_gui.DecorativeWidget

class xsge_gui.**DecorativeWidget**(*parent*, *x*, *y*, *z*, *sprite=None*, *index=None*)
Identical to *Widget*, except that `tab_focus` is `False` by default.

xsge_gui.Label

class xsge_gui.**Label**(*parent*, *x*, *y*, *z*, *text*, *font=None*, *width=None*, *height=None*, *color=None*, *halign='left'*, *valign='top'*)

This widget simply displays some text.

text
The text this label should display.

font
The font this label's text should be rendered with. If set to `None`, the value of *xsge_gui.default_font* is used.

width
The width of the imaginary rectangle the text is drawn in. See the documentation for `sge.gfx.Sprite.draw_text()` for more information.

height
The height of the imaginary rectangle the text is drawn in. See the documentation for `sge.gfx.Sprite.draw_text()` for more information.

halign
The horizontal alignment of the text. See the documentation for `sge.gfx.Sprite.draw_text()` for more information.

valign
The vertical alignment of the text. See the documentation for `sge.gfx.Sprite.draw_text()` for more information.

See the documentation for *xsge_gui.Widget* for more information.

xsge_gui.ProgressiveLabel

class xsge_gui.**ProgressiveLabel**(*parent*, *x*, *y*, *z*, *full_text*, *font=None*, *width=None*, *height=None*, *color=None*, *halign='left'*, *valign='top'*, *rate=1000*)

This widget is a version of *xsge_gui.Label* which progressively builds `text` one character at a time, making it look like the text is being typed in real-time.

full_text
The value that `text` progressively becomes.

rate
The rate at which `text` is built in characters per minute.

xsge_gui.ProgressiveLabel Event Methods

In addition to the event methods inherited from *xsge_gui.Widget*, the following event methods are also available:

ProgressiveLabel.**event_add_character**()
Called when a character is added to `text`.

xsge_gui.Button

class xsge_gui.**Button**(*parent*, *x*, *y*, *z*, *text*, *width=None*, *halign='center'*)
This widget contains some text and can be clicked on by the user.

text
The text contained in the button.

width
The width of the button. If set to None, the width is chosen based on the width of the rendered text.

halign
The horizontal alignment of the text. See the documentation for sge.gfx.Sprite.draw_text() for more information.

See the documentation for *xsge_gui.Widget* for more information.

xsge_gui.Button Event Methods

In addition to the event methods inherited from *xsge_gui.Widget*, the following event methods are also available:

Button.**event_press**()
Called when this button is clicked on, or when the Enter key is pressed while this button is selected.

xsge_gui.CheckBox

class xsge_gui.**CheckBox**(*parent*, *x*, *y*, *z*, *enabled=False*)
This widget can be toggled "on" or "off" by clicking on it.

enabled
Whether or not the checkbox is on.

See the documentation for *xsge_gui.Widget* for more information.

xsge_gui.CheckBox Event Methods

In addition to the event methods inherited from *xsge_gui.Widget*, the following event methods are also available:

CheckBox.**event_toggle**()
Called when the state of the checkbox is toggled by the user.

xsge_gui.RadioButton

class xsge_gui.**RadioButton**(*parent*, *x*, *y*, *z*, *enabled=False*)
This widget is mostly like *xsge_gui.CheckBox*, but clicking on it while it is on will not turn it off, and only one radio button can be on at any given time (i.e. enabling one radio button on a window will disable all others on the same window).

See the documentation for *xsge_gui.CheckBox* for more information.

xsge_gui.RadioButton Event Methods

In addition to the event methods inherited from *xsge_gui.Widget*, the following event methods are also available:

RadioButton.**event_toggle**()
Called when the state of the radiobutton is toggled by the user.

xsge_gui.ProgressBar

class xsge_gui.**ProgressBar**(*parent*, *x*, *y*, *z*, *width=128*, *progress=0*)
This widget displays a bar which can be used to show progress (e.g. of some task being done).

width
The width of the progress bar.

progress
The progress indicated by the progress bar as a factor (i.e. `0` is no completion, `1` is full completion, and `0.5` is half completion).

xsge_gui.TextBox

class xsge_gui.**TextBox**(*parent*, *x*, *y*, *z*, *width=32*, *text=''*, *text_limit=1000*)
This widget provides a place for the user to enter text.

width
The width of the text box.

text
The text in the text box.

text_limit
The maximum number of characters allowed in the text box.

See the documentation for *xsge_gui.Widget* for more information.

xsge_gui.TextBox Event Methods

In addition to the event methods inherited from *xsge_gui.Widget*, the following event methods are also available:

TextBox.**event_change_text**()
Change text event.

Called when the user changes the text in the textbox.

xsge_gui.MenuItem

class xsge_gui.**MenuItem**(*parent*, *x*, *y*, *z*, *sprite_normal=None*, *sprite_selected=None*)
This widget has two sprites: one for when it is selected, and one for when it is unselected. Meant to be used with *xsge_gui.MenuWindow* or *xsge_gui.MenuDialog*.

sprite_normal
The sprite to use as `sprite` when this widget is unselected.

sprite_selected
The sprite to use as `sprite` when this widget is selected.

See the documentation for *xsge_gui.Widget* for more information.

CHAPTER

TWO

XSGE_GUI FUNCTIONS

xsge_gui.**init**()

Prepare this module for use. This function in particular creates the sprites and fonts it uses for windows and widgets. Because of this, it must not be called until after a `sge.dsp.Game` object has been created.

xsge_gui.**show_message**(*parent=None*, *message=''*, *title='Message'*, *buttons=('Ok',)*, *default=-1*, *width=320*, *height=None*)

Show a message and return the button pressed.

Arguments:

- `parent` – The parent handler of the *xsge_gui.MessageDialog* object created. Set to `None` to create a new handler and then destroy it after the dialog is shown.
- `message` – The message shown to the user.
- `title` – The window title of the *xsge_gui.MessageDialog* object created.
- `buttons` – A list of strings to put inside the buttons, from left to right.
- `default` – The index of the default button selected by the keyboard (i.e. the default choice).
- `width` – The width of the *xsge_gui.MessageDialog* object created.
- `height` – The height of the *xsge_gui.MessageDialog* object created. If set to `None`, set the height automatically based on the space needed for the text.

Value returned is the index of the button pressed, where `0` is the leftmost button, or `None` if no button was pressed (i.e. the close button on the window frame was pressed instead).

See the documentation for *xsge_gui.MessageDialog* for more information.

xsge_gui.**get_text_entry**(*parent=None*, *message=''*, *title='Text Entry'*, *text=''*, *width=320*, *height=None*)

Return text entered by the user.

Arguments:

- `parent` – The parent handler of the *xsge_gui.MessageDialog* object created. Set to `None` to create a new handler and then destroy it after the dialog is shown.
- `message` – The message shown to the user.
- `title` – The window title of the *xsge_gui.TextEntryDialog* object created.
- `text` – The text in the text box by default.
- `width` – The width of the *xsge_gui.TextEntryDialog* object created.
- `height` – The height of the *xsge_gui.TextEntryDialog* object created. If set to `None`, set the height automatically based on the space needed for the text.

Value returned is the text entered if the "Ok" button is pressed, or `None` otherwise.

See the documentation for *xsge_gui.TextEntryDialog* for more information.

xsge_gui.**get_menu_selection**(*x*, *y*, *items*, *parent=None*, *default=0*, *font_normal=None*, *color_normal=None*, *font_selected=None*, *color_selected=None*, *background_color=sge.gfx.Color("#00000000")*, *height=None*, *margin=0*, *halign='left'*, *valign='top'*)

Show a menu and return the index of the menu item selected.

Arguments:

- `parent` – The parent handler of the `xsge_gui.TextMenuDialog` object created. Set to `None` to create a new handler and then destroy it after the dialog is shown.
- `default` – The index of the item to select by default.

See the documentation for `xsge_gui.MenuDialog.from_text` for more information.

X

B

C

D

E

F

G

H

I

J

K

L

M

Y

Z