ogSvgWidgets

Components for an svgEditor

Programmer's guide

Widgets and Pickers

Version	Date	Writer	Comments
1.0.00	Thursday, 28 January 2021	OG	first version, Master Candidate
1.5.00	Sunday, 27 June 2021	OG	Add wos_getSize, and a full layout system. This allows to change dynamically the size of the widgets. Add a ruler widget.
1.5.01	Thursday, 15 July 2021	OG	Add the property "is_visibility" on how to manage palettes.
1.5.10	Monday, 13 March 2023	OG	Finalisation and definitive name

Overview

The *ogSvgWidget* is a component as a pool of widgets to bury often used objects and to avoid to see all the "logic" behind and focus on the utilisation, with options and parameters. This approach is very efficient and drives the way you code, gives very homogeneous interfaces. At last, updates but also improvements are supposed to be very less sensitive in the host database.

ogSvgWidget is designed in an extremely regular and generic way. In this sense, most methods can be used for all widgets and pickers : getters, setters are the same. Here you will find all the Widgets and Pickers residing in this components.

In that way, all the generic methods to access the widgets are detailed in the generic documentation "ogTools suite - Widgets common".

from v19 R6, for project mode

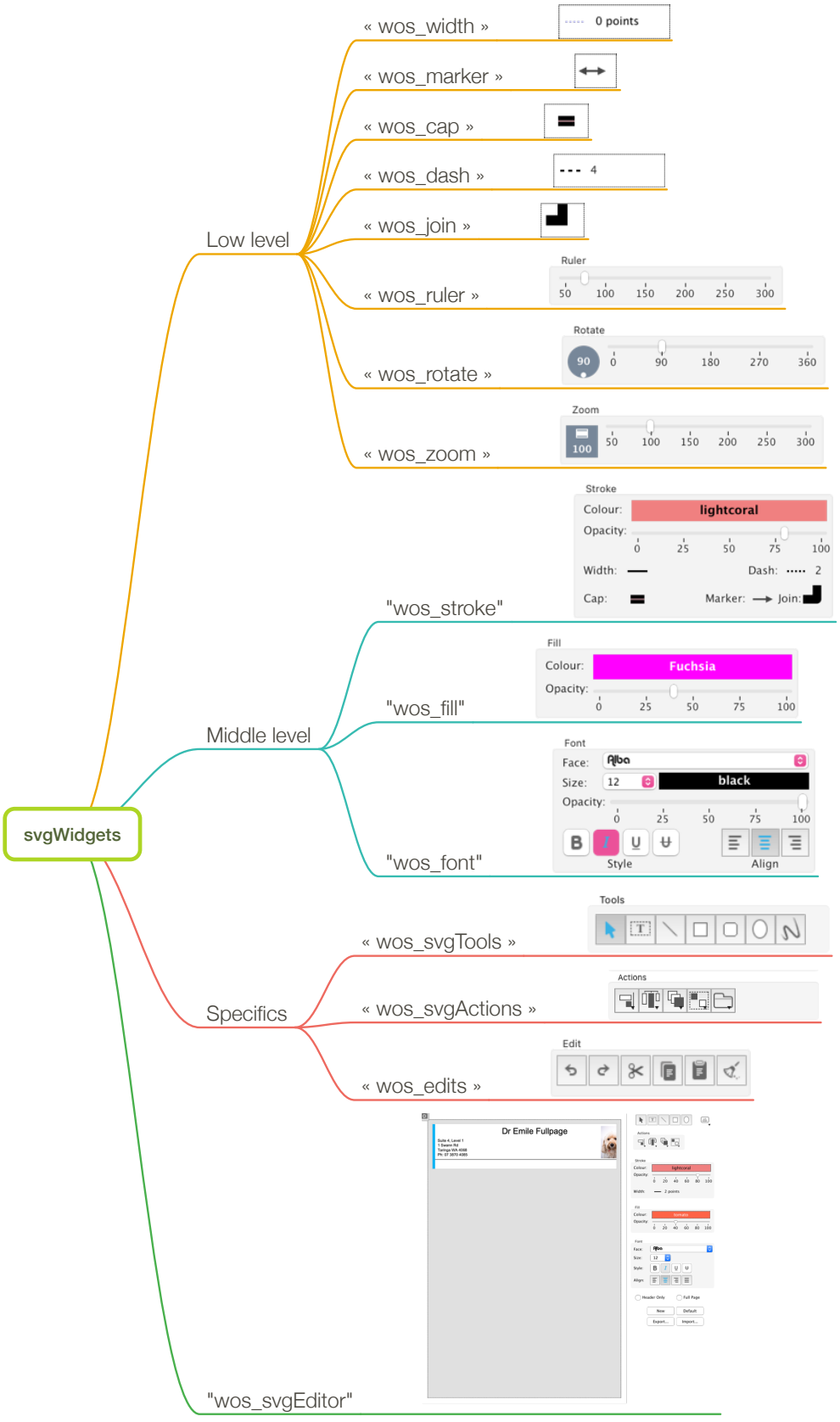
Protée
IT, app design

Protée sarl
31, rue Sainte-Marthe
31000 TOULOUSE

Tél : +33 (0)970 46 56 46
Mob : +33 (0)6 3718 5941
www.protee.org / info@protee.org

Introduction

Here are the actual Widgets and Pickers available in *ogSvgWidgets*.



Object parameters

wos__storage_prefs

Description

All components use one interprocess object as default values for each host created instance. This main interprocess object is accessible with this method.

wos__storage_prefs -> vJ			
Parameter	**Type**		**Description**
vJ	object	↩	Gives back the main prefs object of ogSvgWidgets

Property	Type	Default	Contents
t_name	text	ogSvgWidgets	Component name
t_version	text	x.y.zz	Current version
t_lip_app	text	wos	License app name
l_top	number	100	pixels from top as non usable area for Form windows
al_windowType	collection		“Window type” array: Plain form window; Sheet form window; Pop up form window; Palette form window
l_windowType	number	3	Default window type
t_popup_idle	text	Idle	Label in popup for the “Idle” state.
t_font_default	text	"Lucida Grande"; "Segoe UI"	Current default font
r_fontOffset_coef	number	0.7	Default colour widget in mosaic (else Listbox)
l_svg_scale	number	1	Scale for svg display. You can use 2 or 4 for retina screen.
ui	object		See below

wos__storage_prefs_ui

Description

This is a shortcut of “wos__storage_prefs.ui”.

wos__storage_prefs_ui -> vJ			
Parameter	**Type**		**Description**
vJ	object	↩	Gives back the main prefs.ui object of ogSvgWidgets

Property	Type	Default	Contents
widgets_margin	number	4	Margin between widgets in svgEditor
Editor options	======	======	
selection_fill	text	#666666	Selection fill color
selection_fill_opacity	text	0.1	Opacity
selection_stroke	number	#EEEEEE	Selection stroke color
selection_stroke_opacity	collection	0.7	Opacity
selection_stroke_width	number	1	Width in pixels
Handles	======	======	
handle_radius	number	4	handles squares "radius"
handle_fill_opacity	number	0.7	Opacity
handle_fill	text	#9999FF	Fill color
handle_stroke	text	#EEEEEE	Stroke color
handle_stroke_opacity	number	0.7	Opacity
handle_stroke_width	number	0.5	
Polylines handles	======	======	
line_handle_fill	text	#FF6600	
line_handle_stroke	text	#FF6600	
For "Reserved" rectangles	======	======	
paper_fillColour	text	white	
margins_strokeColour	text	lightgrey	
margins_fillColour	text	none	
reserved_strokeColour	text	grey	
reserved_fillColour	text	lightgrey	

wos__storage_widgets

Description

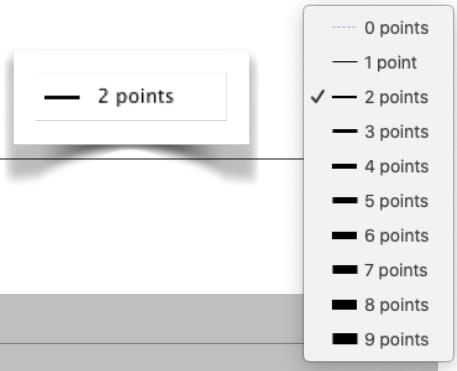
All components use this storage object as default values for each created instance. This main object is accessible with this method. Properties are the default values for widget/pickers. They are default values and are listed here:

wos__storage_widgets -> vJ			
Parameter	**Type**		**Description**
vJ	object	↩	Gives back the main widgets object of ogSvgWidgets

Property object	Widget/Picker
width	A popup for width selection.
marker	A popup to choose marker at start and end of lines, poly-lines…
cap	A popup for cap value
dash	A popup for dash value
join	A popup for join value
ruler	A ruler picker with a yellow label indicator
rotate	A rotation Picker made of a ruler and a rotate display.
zoom	A zoom Picker made of a ruler and a zoom display.
stroke	Made of a colour picker, an opacity ruler, the width picker, and an optional marker picker.
fill	Made of wos_colourPicker and an opacity ruler
font	Made of a font popup, a size popup, an optional colour picker, a style checkboxes and align radios.
svgTools	svgTools picker for the wos_svgEditor.
svgActions	svgActions picker for the wos_svgEditor.
svgEdits	svgEdits picker for the wos_svgEditor.
svgEditor	The main svgEditor!

SVG
Widgets

“wos_width”

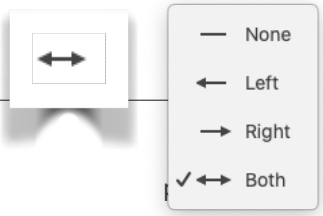


Description

This picker allows you to select a width for line, polyline, rect, circle, etc.

Property widthPicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
width	num	↔	Width of the line

“wos_marker”



Description

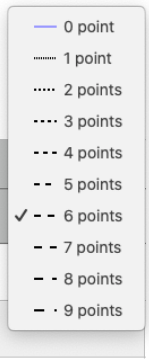
This picker allows you to select the markers to apply at both extremities for line,

Property markerPicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
marker	num	↔	Marker of the line. 0 none, 1 left, 2 right, 3 both.

“wos_dash”

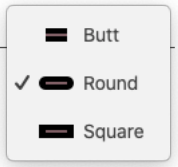
Description

This picker allows you to select the dash option to apply for line, polyline, rect, circle, ellipse.



Property markerPicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
dash	num	↔	Dash of the line in pixels. 0 none, 1 - 9.

“wos_cap”



Description

This picker allows you to select the cap option to apply for line, polyline, rect, circle, ellipse.

Property markerPicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
cap	num	↔	Cap from 0 -2 -> “butt”, “round”, “square”.

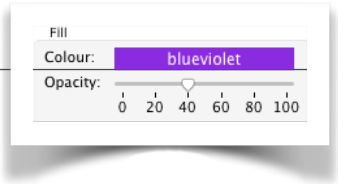
“wos_join”

Description

This picker allows you to select the join option to apply for polyline, rect. ellipse.

Property markerPicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
join	num	↔	Join from 0 - 2 -> “miter”, “round”, “bevel”.

“wos_fill”

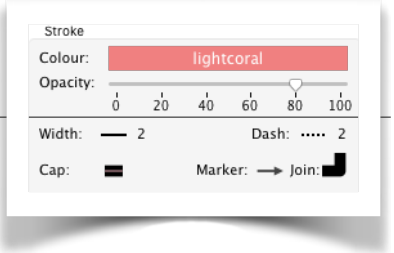


Description

This picker allows you to select the two parameters for the fill of svg objects. The colour object is the resized element.

Property fillPicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
is_displayText	boolean	→	Display the colour name if true. Then the colour is a rectangle filling the width of the widget, else it is a square.
colour	text	↔	Combined colour, can be an colour name, or in the form “rgb(r, g, b)”
opacity	num	↔	Opacity value from 0 to 100

“wos_stroke”

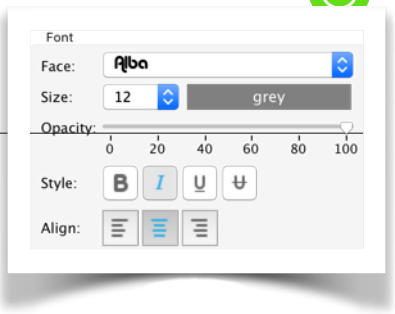


Description

This picker allows you to select the parameters for the stroke of svg objects. The colour object is the resized element.

Property strokePicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
is_displayText	boolean	→	Display the colour name if true
colour	text	↔	Display the colour name if true. Then the colour is a rectangle filling the width of the widget, else it is a square.
opacity	num	↔	Opacity value from 0 to 100
width	num	↔	Width for the included widthPicker
is_dash	boolean	→	If true, the “dash” object is present in the stroke Picker.
dash	num	↔	Value for the included dash picker
is_marker	boolean	→	If true, the marker object is present in the stroke Picker.
marker	num	↔	Marker for the included markePicker
is_cap	boolean	→	If true, the “cap” object is present in the stroke Picker.
cap	num	↔	Value for the included capPicker
is_join	boolean	→	If true, the “join” object is present in the stroke Picker.
join	num	↔	Value for the included joinPicker

“wos_font”

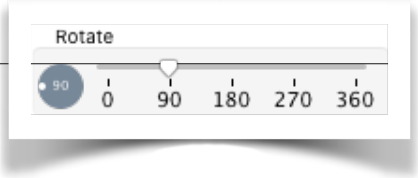


Description

This picker allows you to select the two parameters for the fill of svg objects.
The inside objects are resized only horizontally.

Property fontPicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
is_colour	boolean	→	Display the colour object if true
is_displayText	boolean	→	Display the colour name if true
colour	text	↔	Display the colour name if true. Then the colour is a rectangle filling the width of the widget, else it is a square.
opacity	num	↔	Opacity value from 0 to 100
face	text	↔	Font face
size	num	↔	Font size
style	num	↔	Font style: bits 0 to 3 for Bold, Italic, Underline, Strike
align	num	↔	Alignement: 0 left, 1 center, 2 right

“wos_rotate”

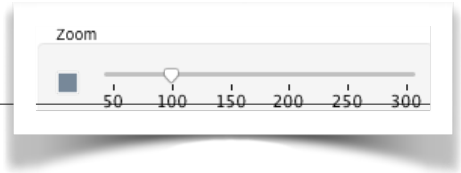


Description

This picker allows you to select a rotation parameter.
The picker is resized both horizontally and vertically. A click on the graphic, or a right click on the ruler propose useful values.

Property rotatePicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
colour	text	→	Colour for the embedded graphical indicator.
type	num	→	Type of graphical indicator (square, circle…)
rotate	num	↔	Rotate value from 0 to 360°

“wos_zoom”

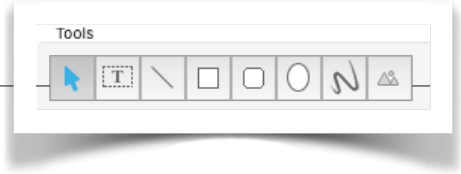


Description

This picker allows you to select a zoom parameter.
The picker is resized both horizontally and vertically. A click on the graphic, or a right click on the ruler propose useful values.

Property zoomPicker			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
colour	text	→	Colour for the embedded graphical indicator.
type	num	→	Type of graphical indicator (square, circle…)
zoom	num	↔	Zoom value from 50 to 300x

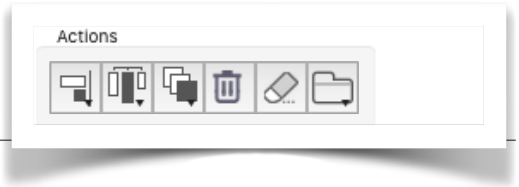
“wos_svgTools”



Description

This picker allows you to select a tool for the svgEditor.
The picker is resized both horizontally and vertically. A sift click on a button changes to STICKY mode.

Property svgTools			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
is_sticky	boolean	↔	False, non sticked by default, sticked with Right click. True, opposite.
tool	text	↔	tool selected. Values are SELECT, TEXT, LINE, RECT, ROUND, CIRCLE, FREELINE, IMG. A right click on a tool makes it “STICKY” indicated by the boolean and also in the tool adding “#STICKY” to the tool value. SELECT, TEXT, IMG can’t be STICKY.
is_sticked	boolean	↔	If the button is sticked.



“wos_svgActions”

Description

This picker allows you to select an action for the svgEditor.
The picker is resized both horizontally and vertically.

Property svgTools			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
count	num	→	Give the count of selected svg item. It will enable buttons depending on value (Align, distribute, group >=2, layer >=1).
mask	num	→	Mask to display the buttons, if 1 in this order msb to lsb : File, Clear, Trash, layer, group, distribute, align : default %1111011 (no group button)
action	text	←	Action chosen.

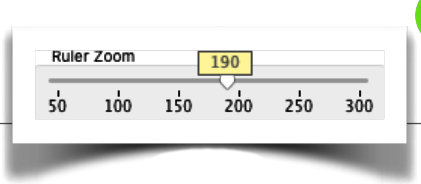
“wos_svgEdits”

Description

This picker allows you to select the standard “Edit” actions.
The picker is resized both horizontally and vertically.

Property svgTools			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
mask	num	→	Mask to display the buttons, if 1 in this order msb to lsb : Clear, Paste, Copy, Cut, Redo, Undo. Default %111111 (all buttons)
enablers	object	→	Give the enabled state for each buttons: is_undo, is_redo, is_cut, is_copy, is_paste, is_clear.
edit	text	←	Edit action chosen.

“wos_ruler”

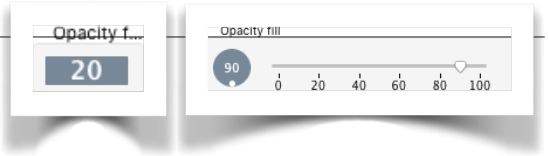


Description

This picker allows you to create a ruler, with an on over yellow tip displaying the value. The picker is resized both horizontally and vertically.

Property svgTools			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
tip	num	→	If empty, display the label, else this tip.
min	num	→	min value of the ruler
max	num	→	max value of the ruler
unit	num	→	unit for graduations
step	num	→	step for the values
ruler	num	↔	Value in and out

“wos_opacity”



Description

This picker allows you to select a standard opacity value.
The picker is resized both horizontally and vertically. It has some layouts based on its host’s size.

Property svgTools			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
label	text	→	If empty, no group is displayed, else this is the label of the group box.
colour	text	→	colour for the left svg displayer.
mode	text	→	Give the mode for the right click menu “stroke”, “fill”, “font”.
opacity	num	↔	Value in and out

SVG
Editor widget

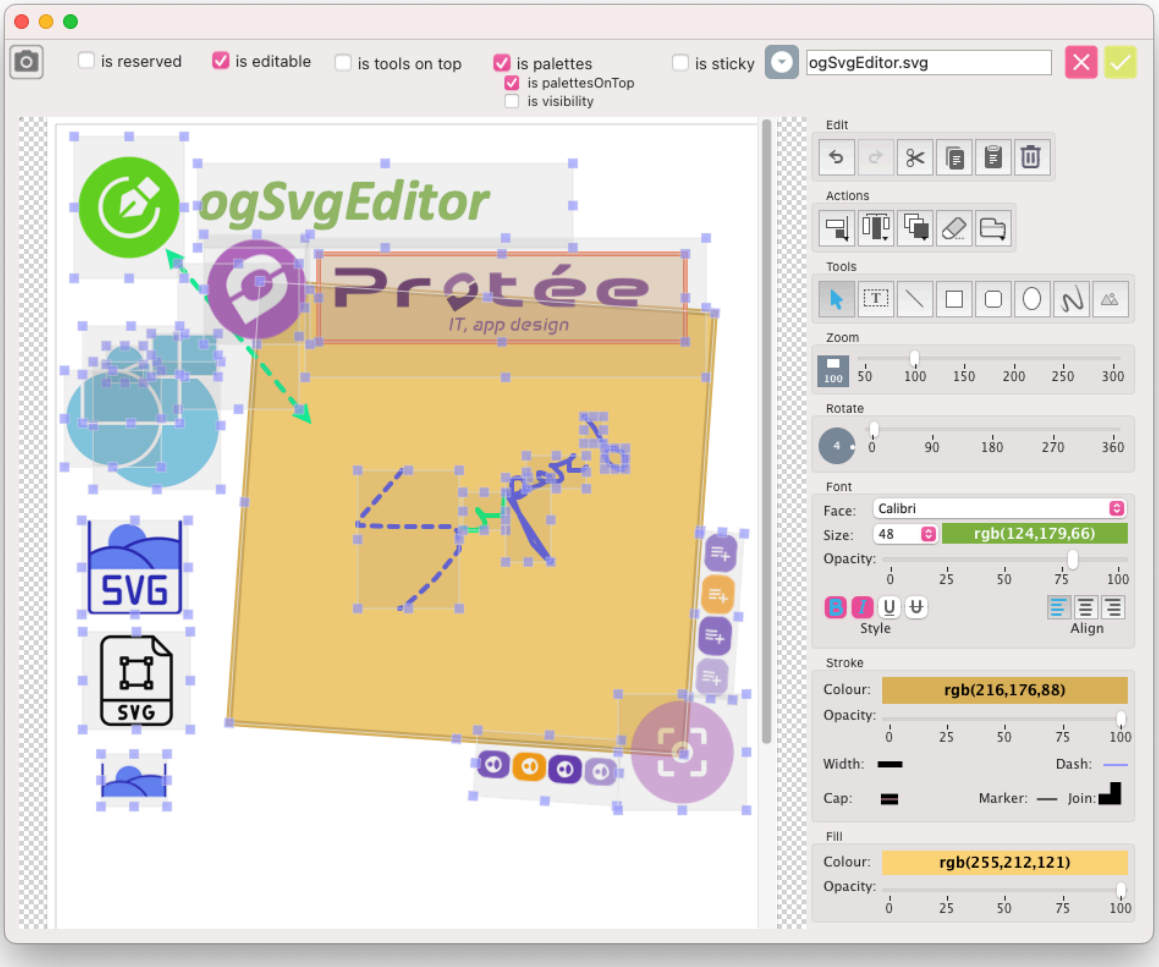
“wos_svgEditor”

Description

This picker allows you to edit an svg file.
The picker is resized both horizontally and vertically. This is a high level component, with some dedicated methods to manage it. This Component deals with svg content (text) and manage by itself the dom, and to avoid any memory leaks.
Only those elements are imported and edited: "textArea", "line", "rect", "circle", "ellipse", "polyline", “image".

Property svgEditor			
property	**Type**		**Description**
is_editable	boolean	→	If true, the widget is editable.
is_modified	boolean	←	True if the initial picture has been modified.
is_toolsOnTop	boolean	→	True to get palettes Tools and Actions on Top of svgEditor.
is_palettes	boolean	→	True to display palettes. Zoom is always at the right.
is_palettesOnTop	boolean	→	True to display palettes on top. Only used with is_palettes
is_visibility	boolean	→	How to manage palettes. True stacked with visibility, false by disabled.
is_domToClose	boolean	→	Default True. If false, the dom closing is not managed by the widget. This allows you to use this widget in a Form page >1. Use method “wos_svgEdit_dom_close” to do so.
is_spaces	boolean	→	True to manage spaces informations: in colour picker, and right click.
is_marker	boolean	→	True to manage stroke’s marker information.
is_cap	boolean	→	True to manage stroke’s cap information.
is_join	boolean	→	True to manage stroke’s join information.
is_sticky	boolean	↔	False, non sticked by default, sticked with Right click. True, opposite.
paper • width • height	object	→	Gives the paper size in pixels. Properties width and height.
margin • left • top • right • bottom	object	→	Gives the margin inside the paper in pixels. Properties left, top, right, bottom.
reserved • left • top • width • height	object	→	Gives the position in pixels of a rectangle drawn to indicate where not to draw. Relative to area (paper-margins). Put 0 in width to have it off.

The svg zone's width is the widget's width minus 260 pixels for all the palettes and minus 10 pixels gap. The svg zone's height is the widget's height. But the minimum height of the svgEditor's widget instance must be 650 pixels, to allow it to display all its palettes.
The component provides a method to correctly set all dimensions for trivial situations.
See below "wos_svgEdit_paper_calculate".



“svgEditor_ui” object

Description

There are some global parameters you can change to define the UI. See below access thru:
$ob:=**wos__storage_prefs**.ui

Property svgEditor_ui		
property	**Type**	**Description**
selection_fill	text	Selection rectangle fill colour
selection_fill_opacity	num	Selection rectangle fill opacity (0-1)
selection_stroke	text	Selection rectangle stroke colour
selection_stroke_opacity	num	Selection rectangle stroke opacity (0-1)
selection_stroke_width	num	Selection rectangle stroke width
handle_fill	text	Handles fill colour
handle_fill_opacity	num	Handles fill opacity (0-1)
handle_stroke	text	Handles stroke colour
handle_stroke_opacity	num	Handles stroke opacity (0-1)
handle_stroke_width	num	Handles stroke width
handles_radius	num	Radius for the handles rectangle
lineHandle_strokeColour	text	Stroke colour for the handles
lineHandle_fillColour	text	Fill colour for the handles
margins_fillColour	text	Fill colour for the margins rectangle
margins_strokeColour	text	Stroke colour for the margins rectangle
paper_fillColour	text	Fill colour for the paper main rectangle
reserved_fillColour	text	Fill colour for the reserved rectangle
reserved_strokeColour	text	Stroke colour for the reserved rectangle
widgets_margin	num	Vertical margin between widgets

Methods

“wos_svgEdit_get_image”

Description

This method gives you back a picture based on the svg edited in a widget.
picture:=wos_svgEdit_get_image ({svgEditor})

Params	Type		Description
svgEditor	text	→	Choose the source widget, itself if not given.
picture	picture	←	The picture given back

“wos_svgEdit_get_svg”

Description

This method gives you back a picture based on the svg edited in a widget.
svg_txt:= wos_svgEdit_get_svg ({svgEditor})

Params	Type		Description
svgEditor	text	→	Choose the source widget, itself if not given.
svg_txt	text	←	The svg data in text

“wos_svgEdit_set_image”

Description

This method sets the image in a widget with a picture.
wos_svgEdit_set_image (picture ; {svgEditor})

Params	Type		Description
picture	picture	→	The picture with a dom
svgEditor	text	→	Choose the source widget, itself if not given.

“wos_svgEdit_set_svg”

Description

This method sets the image in a widget with a picture.
wos_svgEdit_set_svg (svg_txt ; {svgEditor})

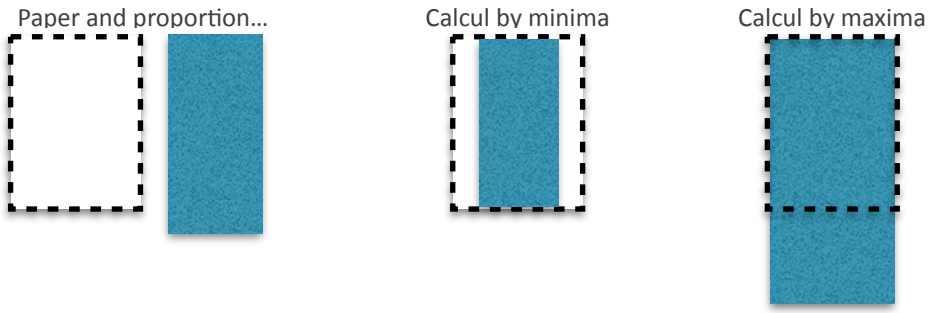
Params	Type		Description
svg_txt	text	→	The svg data in text
svgEditor	text	→	Choose the source widget, itself if not given.

“wos_svgEdit_paper_fit”

Description

This method gives you back the size in pixel for the proportion you want in the current widget’s canvas area.
The result is by minima. If you put a negative value for the proportion, the result is by maxima.
wos_svgEdit_paper_fit (paper_factor ; ptr_width ; ptr_height ; {svgEditor})

Params	Type		Description
paper_factor	real	→	For the case of all margins the same. Put the Width/Height you wish, value from 0 to 1. By minima, and by maxima if negative value.
ptr_width	pointer	←	pointer on a longint for width.
ptr_height	pointer	←	pointer on a longint for height.
{svgEditor}	text	→	Choose the source widget, itself if not given.



“wos_svgEdit_paper_calculate”

Description

This method will setup the object of an “svgEditor’, for the object properties “paper”, “margin”, “reserved”. This is a shortcut to those parameters for trivial situations.
wos_svgEdit_paper_calculate (vJ_svgEditor ; paper_width ; paper_height ; {margin_factor} ; {header_factor}; {footer_factor})

Params	Type		Description
vJ_svgEditor	object	→	svgEditor object
paper_width	longint	→	The paper width.
paper_height	longint	→	The paper height.
margin_factor	real	→	For the case of all margins the same. Put a percent of Width, and -1 to keep unchanged. Value from 0 to 1.
header_factor	real	→	Header percent, from 0 to 1. Default 1.
footer_factor	real	→	Footer percent, from 0 to 1. Default 0.

“wos_svgEdit_get_canvasSize”

Description

This method gives you back the actual calculated width and height of the drawing area (canvas) inside the given svgEditor widget.

wos_svgEdit_get_canvasSize (->width ; ->height ; {svgEditor})

Params	Type		Description
ptr_width	pointer	←	pointer on a longint for width.
ptr_height	pointer	←	pointer on a longint for height.
svgEditor	text	→	Choose the source widget, itself if not given.

“wos_svgEdit_get_widgetSize”

Description

This method gives you back the minimum width and height needed to be sure to get the proper drawing area (canvas) for the given svgEditor widget. This will use the paper size properties, and the “is_toolsOnTop” and “is_palettes” properties.
The given size is for the svgEditor widget’s instance, and assume you will have no scrollbars for the canvas, and that the palettes will have enough space to be drawn.

wos_svgEdit_get_widgetSize (->width ; ->height ; {svgEditor})

Params	Type		Description
ptr_width	pointer	←	pointer on a longint for width.
ptr_height	pointer	←	pointer on a longint for height.
svgEditor	text	→	Choose the source widget, itself if not given.

“wos_svgEditor”

Description

This method opens an svgEditor window form.
isOk:=**wos_svgEditor** (vJ_option)

Params	Type		Description
vJ_option	object	→	Options object. All properties overwrite the default values defined.
isOk	longint	←	True, the form has been accepted.

“wos_svgEdit_dom_close”

Description

This method close an eventual existing DOM for this widget.
isOk:=wos_svgEdit_dom_close (vJ_svgEditor)

Params	Type		Description
vJ_svgEditor	object	→	svgEditor object
isOk	longint	←	True, a dom has been closed

An alternative of using this method is to manually close an eventual dom locking at the object, property domRef.
And execute : DOM CLOSE XML($ob.domRef)

Host method wrapper
“wosh_picLib_callback”

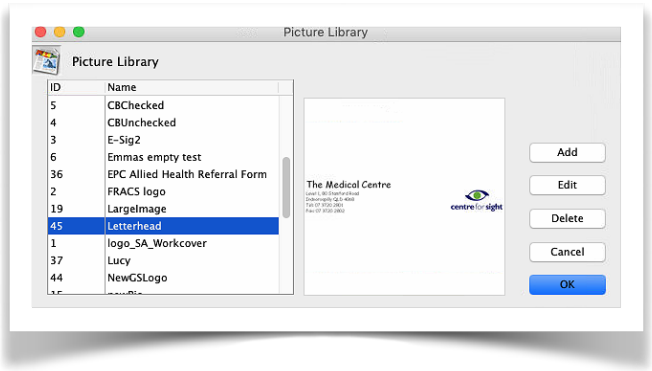
Description

This method is called when you click on the “Picture” tool’s button. This is a wrapper that must exist in the Host, with the attribute “Shared”. If this method is not found, the standard file “open dialog” is proposed (to pick up a picture on disk). The purpose is to propose a standard Picture Library manager to suit your needs, ie in Genie, the table [PictureLibrary] and all the methods and Forms that manage it. This method works with an object parameter.

wosh_picLib_callback (ob_picLib)

Params	Type		Description
ob_picLib	object	↔	Object parameters, in and out. Answer “entity” or “entities”.

An example of **wosh_picLib_callback**:




```
C_OBJECT($ob)
$ob:=$1

C_TEXT($vT_query)
$vT_query:=String($ob.query)
Case of
    : ($vT_query="") //
        PICLIB_EditWindow
        If ((OK=1) & ([PictureLibrary]ID#0))
            $ob.entity:=ds.PictureLibrary.query("ID = :1";
[PictureLibrary]ID).first()
        End if
    : ($vT_query="id:@") //
        $ob.entity:=ds.PictureLibrary.query("ID =
:1";Num($vT_query)).first()

    : ($vT_query="all") // Get all table entities
        $ob.entities:=ds.PictureLibrary.all()
End case
```

Implementation

The way you implement the svgEditor widget depends on if you put it on a Form on page 1 or on a page higher than 1.

Page 1 is the most simple, but for pages higher, 4D subForm event manager is just shitty, as every time you land on that page, a On Load event is triggered, and when you leave that page, a On Unload event is triggered too, making the stored Dom being cleared. Here is how to deal with that in order to navigate through pages, and keep the svgEditor consistent with its own datas.

- -On Load event is generated only once, that guarantees you will initialise the widget only once.
- The inside widget On Unload event is disableable, to let the widget keep the Dom not closed.
  - but, in the host On Unload event, you will have to close the Dom by yourself.