Graphical Module

documentation

Presentation

Graphical Module is somehow a **graphical library** designed for creating user interfaces using **SuperCollider IDE**.

It's purpose is to provide a **prettier UI than SC's basic QT implementation**.

Instead of being packaged as a Quark,which adds extra steps when seting up environnement, it takes advantage of SC's global variable system, **creating a global dictionary which contains both UI variables and widget creation functions**.

To get it running, **execute *GM_main.scd* while scide is running**.

You can find a demo at *GM_demo.scd*.

Basically, **all widget are instances of the UserView class**, with custom drawFunc, variables and methods.

Usage

To **initialize** the library, **execute *GM_main.scd***, either by hand, or by using

```
this.executeFile( ( "path/to/GM_main.scd" ).standardizePath ) );
```

Once it's done, **you have access to a global variable, *~gm***, which can be used to setup a Graphical User Interface.

~gm both contains GUI variables and widget creation functions.

To **setup a GUI variable** :

```
~gm.put( \mainColor, Color( 1, 0, 0 ) );
```

Note : this doesn't modify previously instanced widgets.

To **get a widget** :

```
var my_button = ~gm.at( \simpleButton ).value() ;
var my_custom_button = ~gm.at(
	\simpleButton ).value(
	text : « Custom Button »,
	backColor : Color.blue
) ;
```

All widgets have a *bindFunction* method which allows you to **bind it to your own function**. This function’s arguments will depend on the widget :

```
var diamondSlider = ~gm.at( \diamondSlider ).value();

diamondSlider.bindFunction( {
	| value |
	( ‘You slided to ‘ ++ value.asString ).postln;
} );
```

In general, you’ll want **a Window instance and store widgets inside layouts** :

```
(
var win = Window(
	"GM Example",
	Rect(
		100,
		100,
		300,
		150
	)
);

var slider = ~gm.at( \diamondSlider ).value();

slider.bindFunction( { | newValue |
	( "You slided to " ++ newValue.asString ).postln } );

win.layout_(
	VLayout(
		slider
	)
);
```

Now, finally, a little bit of fun, with **server running** :

```
(
var win = Window(
	"GM Example",
	Rect(
		100,
		100,
			300,
			150
		)
);

var slider = ~gm.at( \diamondSlider ).value(
		minVal: 55,
		value: 110,
		maxVal: 880,
		setGrowthType: \exp
);

var synth;

SynthDef( \sine, {
		| out = 0, freq = 110, amp = 0.25 |
		var snd = SinOsc.ar( freq, mul: amp );
		Out.ar( out, [ snd, snd ] )
} ).add;

slider.bindFunction( { | newValue |
		synth.set( \freq, newValue );
		( "You slided to " ++ newValue.asString ).postln } );

win.layout_(
		VLayout(
			slider
		)
);

SystemClock.sched( 0.01, { synth = Synth( \sine ) } );

win.front;
)
```

Variables Overview

Here, *~gm* variables and what they're supposed to do. Access through *~gm.at(\symbol)*, modify with *~gm.put(\symbol, value)*.

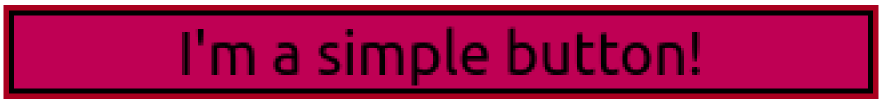
This provides a palette functionnality, although unique. **Changing a variable doesn't modify previously instanced widgets. If you want to use it as a global palette, you'll have to modify variables *before* creating widgets.**

Classes Overview

Here, all classes detailed.

Simple Button

Yeah. Simple button. Click → Trigger.



Instanciation example :

```
var button = ~gm.at(
	\simpleButton ).value(
	backColor: Color.red,
	borderColor: Color.green,
	backgroundColor: Color.blue,
	font: Font.default,
	fontColor: Color.white,
	hasBorderInset: true,
	borderSize: 16,
	text: "Click me !";
) ;
```

Variables :

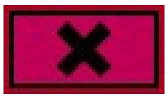
- **backColor** : a *Color*. The color of the button. Default to *~gm.at(\mainColor)*.
- **borderColor** : a *Color*. The color of the border. Default to *~gm.at(\borderColor)*.
- **backgroundColor** : a *Color*. The color of the second border if *hasBorderInset* is *true*. Default to *~gm.at(\backgroundColor)*.
- **font** : a *Font*. The font used to display text. Default to *~gm.at(\mainFont)*.
- **fontColor**: a *Color*. The color of the text. Default to *~gm.at(\fontColor)*.
- **hasBorderInset** : a *Boolean*. Adds a second, inner, border at the button. Default to *~gm.at(\hasBorderInset)*.
- **borderSize** : an *Int*. The size of the borders. Default to *~gm.at(\borderSize)*.
- **text** : a *String*. Displayed text. Default to « text ».

Methods :

- **setText(** ***String*** **)** : sets displayed text.
- **setBorderSize(** ***Int*** **)**: sets border size.
- **setBackColor(** ***Color*** **)** : sets the color of the button.
- **setBorderColor(** ***Color*** **)** : sets the color of the border.
- **setBackgroundColor(** ***Color*** **)** : sets the color of the inner border if *hasBorderInset* is *true* .
- **setInset(** ***Boolean*** **)** : activates or deactivates the inner border.
- **setFont(** ***Font*** **) :** sets the text font.
- **setFontColor(** ***Color*** **)** : set the text color.
- **bindFunction(** ***Function*** **)** : bind a function to be triggered when pressed. No arguments.

Close Button

A ready made close button. Has a cross on it because of cultural evolution. Cross is meant to be transparent and it's color is thus tied to *backgroundColor* . Click → Trigger.



Instanciation example :

```
var closeButton = ~gm.at(
        \closeButton ).value(
        backColor: Color.red,
        borderColor: Color.green,
        backgroundColor: Color.blue,
        hasBorderInset: true,
        borderSize: 16,
        crossWidth: 6;
) ;
```

Variables :

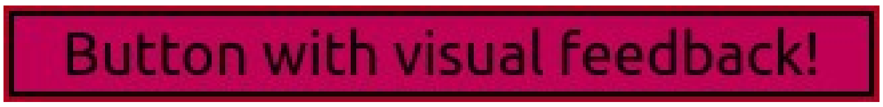
- **backColor** : a *Color*. The color of the button. Default to *~gm.at(\mainColor)*.
- **borderColor** : a *Color*. The color of the border. Default to *~gm.at(\borderColor)*.
- **backgroundColor** : a *Color*. The color of the second border if *hasBorderInset* is *true*. Default to *~gm.at(\backgroundColor)*.
- **hasBorderInset** : a *Boolean*. Adds a second, inner, border at the button. Default to *~gm.at(\hasBorderInset)*.
- **borderSize** : an *Int*. The size of the borders. Default to *~gm.at(\borderSize)*.
- **crossWidth** : an *Int*. The width of the cross lines. Default to 6.

Methods :

- **setBorderSize(*Int*)** : sets border size.
- **setBackColor(*Color*)** : sets the color of the button.
- **setBorderColor(*Color*)** : sets the color of the border.
- **setBackgroundColor(*Color*)** : sets the color of the inner border if *hasBorderInset* is *true* .
- **setInset(*Boolean*)** : activates or deactivates the inner border.
- **setCrossWidth(*Int*)** : sets the cross lines width.
- **bindFunction(*Function*)** : bind a function to be triggered when pressed. No arguments.

Feedback Button

A better button than the simple one, because it has a visual feedback. When clicked, it's back color will transition between feedBack and backColor. It uses default QT 60 FPS setting, but due to inheritance, you can change this with *view.frameRate = Int* . Click → Trigger.



Instanciation example :

```
var feedbackButton = ~gm.at(
        \feedbackButton ).value(
        backColor: Color.red,
        feedbackColor: Color.black,
        borderColor: Color.green,
        backgroundColor: Color.blue,
        font: Font.default,
        fontColor: Color.white,
        hasBorderInset: true,
        borderSize: 16,
        text: "Click me !",
        animationLength = 20;
) ;
```

Variables :

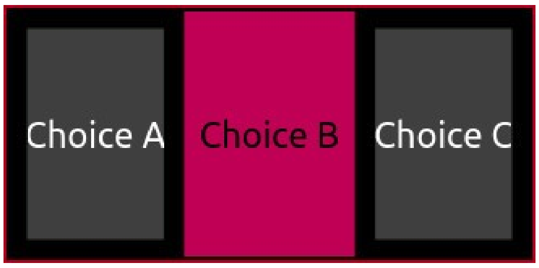
- **backColor** : a *Color*. The color of the button. Default to *~gm.at(\mainColor)*.
- **backColor** : a *Color*. The color the button will transition from to *backColor* when pressed. Default to *Color.black*.
- **borderColor** : a *Color*. The color of the border. Default to *~gm.at(\borderColor)*.
- **backgroundColor** : a *Color*. The color of the second border if *hasBorderInset* is *true*. Default to *~gm.at(\backgroundColor)*.
- **font** : a *Font*. The font used to display text. Default to *~gm.at(\mainFont)*.
- **fontColor**: a *Color*. The color of the text. Default to *~gm.at(\fontColor)*.
- **hasBorderInset** : a *Boolean*. Adds a second, inner, border at the button. Default to *~gm.at(\hasBorderInset)*.
- **borderSize** : an *Int*. The size of the borders. Default to *~gm.at(\borderSize)*.
- **animationLength**: an *Int*. The number of frames the animation lasts. Default to 20.
- **text** : a *String*. Displayed text. Default to « text ».

Methods :

- **setText(** ***String*** **)** : sets displayed text.
- **setBorderSize(** ***Int*** **)** : sets border size.
- **setBackColor(** ***Color*** **)** : sets the color the button will transition from to *backColor* when pressed.
- **setFeedbackColor(** ***Color*** **)** : sets the color of the button.
- **setBorderColor(** ***Color*** **)** : sets the color of the border.
- **setBackgroundColor(** ***Color*** **)** : sets the color of the inner border if *hasBorderInset* is *true* .
- **setInset(** ***Boolean*** **)** : activates or deactivates the inner border.
- **setFont(** ***Font*** **)** : sets the text font.
- **setFontColor(** ***Color*** **)** : set the text color.
- **setAnimationLength(** ***Int*** **)** : sets animation length, in frames. Default QT is 60 FPS.
- **bindFunction(** ***Function*** **)** : bind a function to be triggered when pressed. No arguments. Animation resets when pressed.

Multi Button

A menu containing multiple choices, with only one selection at the time.



Instanciation example :

```
var multiButton = ~gm.at(
	\multiButton ).value(
	labels: [ "Choice A", "Choice B", "Choice C" ],
	backColorSelected: Color.red,
	backColorUnselected: Color( 0.25, 0.25, 0.25 ),
	borderColor: Color.green,
	backgroundColor: Color.blue,
	borderSize: 16,
	font: Font.default,
	fontColorSelected: Color.black,
	fontColorUnselected: Color.black,
	unselectedRatio: 0,75,
	orientation: \horizontal,
	currentState: 0;
) ;
```

Variables :

- **labels** : an *Array* of *Strings*. Will determine the number of choices and their display.
- **backColorSelected** : a *Color*. The color of the currently selected button. Default to *~gm.at(\mainColor)*.
- **backColorUnselected** : a *Color*. The color of all unselected buttons. Default to *Color(0.25, 0.25, 0.25)*.
- **borderColor** : a *Color*. The color of the border. Default to *~gm.at(\borderColor)*.
- **backgroundColor** : a *Color*. The color of the background. Default to *~gm.at(\backgroundColor)*.
- **font** : a *Font*. The font used to display text. Default to *~gm.at(\mainFont)*.
- **fontColorSelected**: a *Color*. The color of the text of the selected button. Default to *~gm.at(\fontColor)*.
- **fontColorUnselected**: a *Color*. The color of the text of all unselected buttons. Default to *Color.white*.
- **unselectedRatio** : an *Float*. The size ratio of the unselected buttons compared to the selected button. Default to 0.8 . Shouldn't be more than 1 . Setting it too low might cause the buttons to be smaller than their text.
- **orientation**: a *Symbol*. Should be either *\horizontal* or *\vertical*. The direction of the menu. Only checks if value is equal to *\horizontal*. Setting any other value will force a vertical orientation. Defaults to *\horizontal*.
- **currentState** : an *Int*. References the current selected button index. Default to 0 .

Methods :

- **setText(** ***String*** **)** : sets displayed text.
- **setBorderSize(** ***Int*** **)** : sets border size.
- **setBackColor(** ***Color*** **)** : sets the color the button will transition from to *backColor* when pressed.
- **setFeedbackColor(** ***Color*** **)** : sets the color of the button.
- **setBorderColor(** ***Color*** **)** : sets the color of the border.
- **setBackgroundColor(** ***Color*** **)** : sets the color of the inner border if *hasBorderInset* is *true* .
- **setInset(** ***Boolean*** **)** : activates or deactivates the inner border.
- **setFont(** ***Font*** **)** : sets the text font.
- **setFontColor(** ***Color*** **)** : set the text color.
- **setAnimationLength(** ***Int*** **)** : sets animation length, in frames. Default QT is 60 FPS.
- bindFunction(*Function*) : bind a function to be triggered when pressed. No arguments. Animation resets when pressed.