1 Installation

The UIPainter project works in 3.0, needed Spec. You can also install it using the following instructions :

```
Gofer new
  smalltalkhubUser: 'ErwanDouaille' project: 'UIPainter';
  configurationOf: 'UIPainter';
  load.

(Smalltalk at: #ConfigurationOfUIPainter) load
```

Otherwise, you can download an image including UIPainter by default, following this link : UIPainter

2 How to use it

You can launch the UIPainter using :

```
UIPainter open
```

or by using the tools menu.

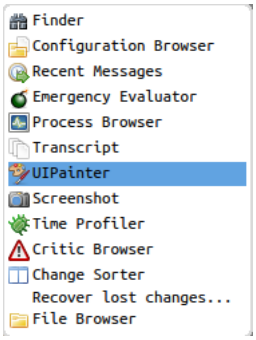


FIGURE 2.1 – UIPainter

The UIPainter project is used to graphically create user interface (UI) by dragging and dropping items in a canvas.

This is the UIPainter interface :

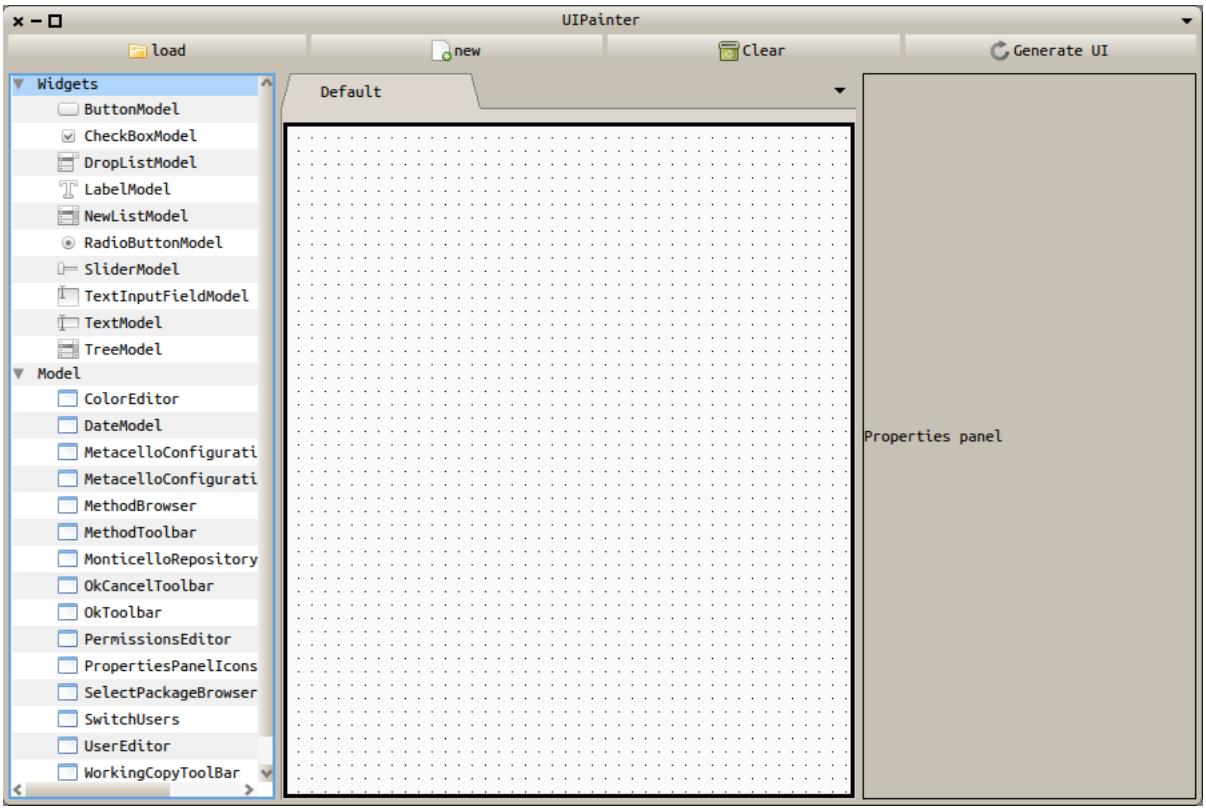


FIGURE 2.2 – UIPainter

2.1 Components

2.1.1 Toolbar

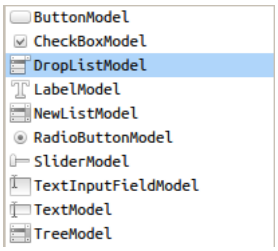
Load : Allow the user to load an existing spec ui and add widgets into the canvas. The user allow the user to modify an existing spec window

New : It open a new tab.

clear : Remove all widgets into your canvas

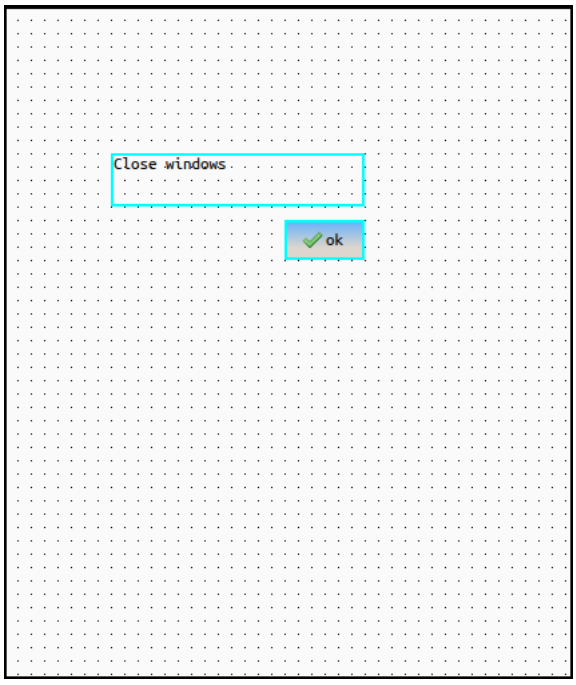
GenerateUI : This is the most important button. It generate the Spec source code corresponding to the canvas contents.

2.1.2 Widget list



This panel provides Spec widgets. Each item can be dropped into the canvas

2.1.3 Canvas



This is the place where you drag and drop items.

2.1.4 Properties panel

Here is the place where you can settup widgets. To make it appears, you have to choose a widget. When you finish to setup the widget, you have to apply settings by clicking on "Apply settings" button.

2.1.5 Halo

close :  close and delete the widget

duplicate :  duplicate the widget, including properties

move :  move the widgets :)

sticky :  active or disable it. When enabled, the morph will stick to the closest grid point. The color will change depending of the activation state

resize :  resize the widgets.

3 Classes

3.1 Related to the user interface

UIPainter is the main class. It's a ComposableModel extended class which implement :

- UIPainterToolbar
- UIPainterPanel
- UIPainterWidgetLister
- UIPainterPropertiesPanel

3.2 Related to the properties panel

All related classes are in UIPainter-Core-Properties. Use case in Add new items chapter.

3.3 Related to the Spec

Spec classes are in the UIPainter-Core-Spec. There is 2 class, UIPainter-SpecGeneration and UIPainterSpecImporter

3.3.1 Spec generation

```
UIPAinterSPecGeneration>>generate: aPanel
```

Is the entry point of the class. Panel have to be an UIPainterPanel instance. You also have to specify the class name, the category and the title name of the generated UI.

You can see the use case :

```
specGenerator := UIPainterSpecGeneration generate: panel.
  specGenerator
    classname: self getClassName;
    category: self getCategoryName;
    title: self getTitleName;
    generate
```

3.3.2 Spec importation

```
UIPAinterSPecImporter>>import: aComposableModel inPanel: aPanel
```

Is the entry point of the class. Panel have to be an UIPainterPanel instance and aComposableModel a subclass of ComposableModel.

3.4 Related to the Widgets

All widgets related classes are in UIPainter-Core package.

WidgetWrapper : is the object containing efficient widgegt informations. Properties, spec related model, id, morph representation ...

WidgetWrapperMorph : this is the morph representation used in UIPainter canvas.

WidgetWrapperMorphHalo : is the halo used for UIPainter widgets.

4 Add new items

If you want to add new widgets into the widget list you have to implement this methods into class side :

4.1 Widgets

```
YourClass>>icons
  "the icon use in the widget list. This is the default icon"
  ^ ThemeIcons new smallWindowIcon

YourClass>>uiPainter
  "use in UIPainter ? return true or false"
  ^true
```

Of course, your class should extend from ComposableModel
Another thing to implement is :

```
YourClass>>properties
  ^ Dictionary new
    at: #text: put: #String;
    at: #whenTextChanged: put: #BlockClosure;
    yourself.
```

properties have to return a Dictionary containing a selector of your class and the expected object. This is an example from LabelModel.

If you didn't implement properties nothing happends and it means than your widget will not be parameterized from the UIPainter.

4.2 PropertiesPanel

Adding a properties panel for a specific Object is really easy. I added an example for the Integer class.

```
Integer>>uipainterPropertiesPanel
  ^ #PropertiesPanelInteger
```

PropertiesPanelInteger is the properties panel class. Your properties panel have to extend from PropertiesPanel. Your panel have to implement two methods

```
YourPanel>>readContents
  ^ Association
    key: input text contents asString
    value:
      (nil class compiler
        evaluate: (ReadWriteStream on: input text contents from: 1 to: input text
    contents size)
          in: nil
          to: nil
          notifying: nil
          ifFail: [ ]
          logged: true)
```

readContents is the default api for the spec generation to retrieve input data. It have to return an association containing as the key, the string input data, and as the value, the compiled input data, meaning the expected object.

```
YourPanel>>setContents: anObject
  anObject isNil ifTrue: [ ^ self ].
  input text: anObject key asString
```

setContents : anObject is the way you will setup your panel by retrieving old input data. This object is an Association.

PropertiesPanelInteger contains a text field object. Of course you have to setup #initializeWidgets and class side #defaultSpec