CS 535 Object-Oriented Programming & Design
Fall Semester, 2001
Doc 17 Simple GUI Tutorial
Contents

Reference & Reading

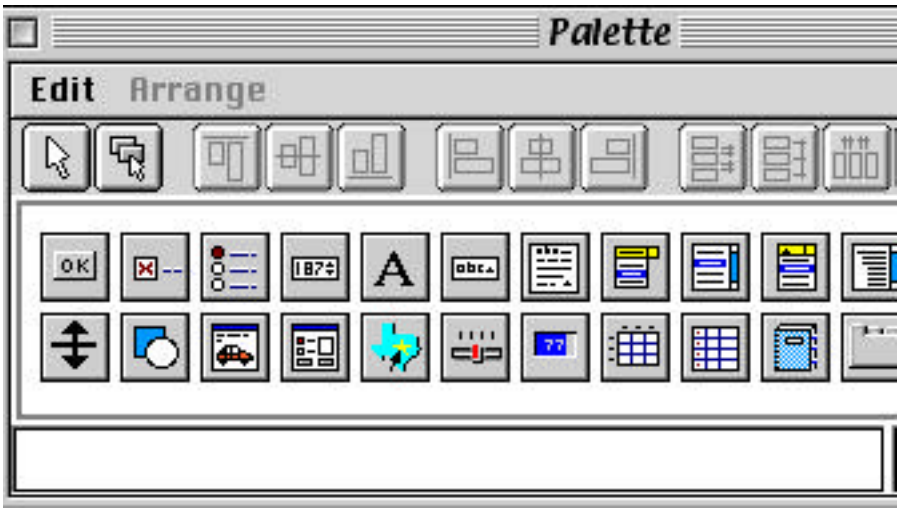
VisualWorks ***GUI Developer's Guide*** (GUIDevGuide.pdf) Chapters 1 & 2

Simple GUI Tutorial

GUIs in VisualWorks are created using the UIPainter. Open the UIPainter by clicking on the icon:  in the Launcher.

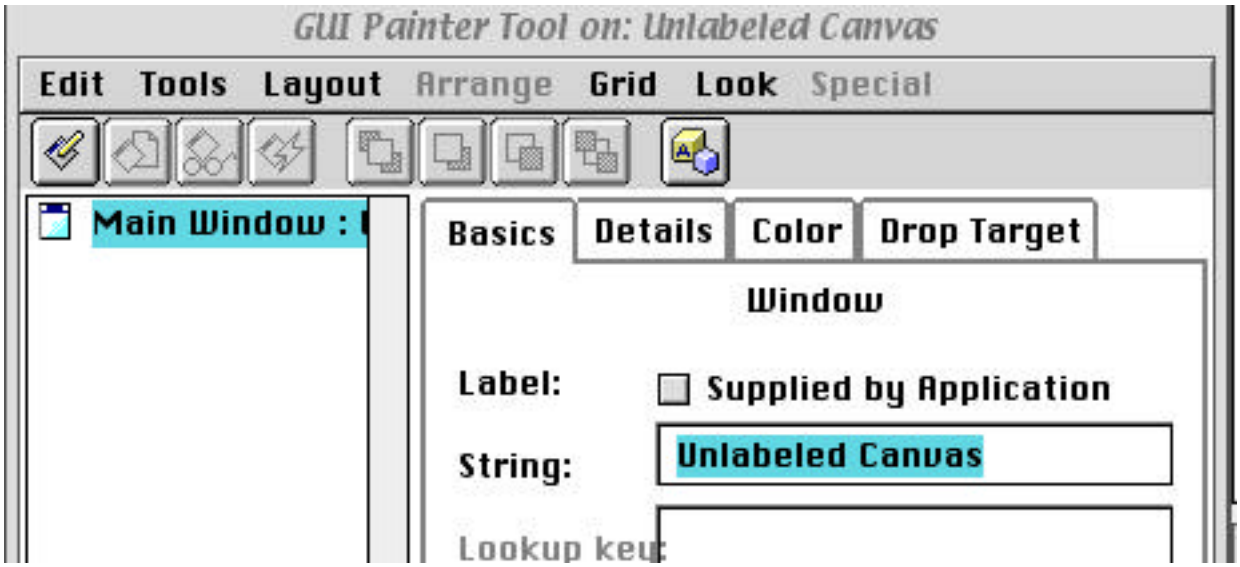
The UIPainter has three parts: GUI Painter Tool, a canvas and a palette.

Palette



Canvas

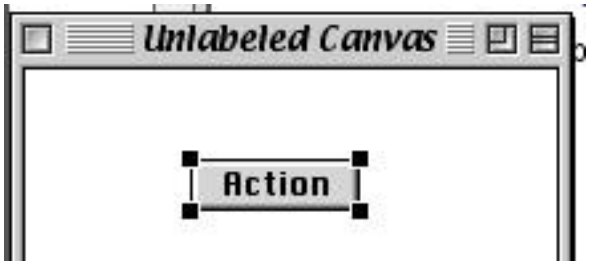
GUI Painter Tool



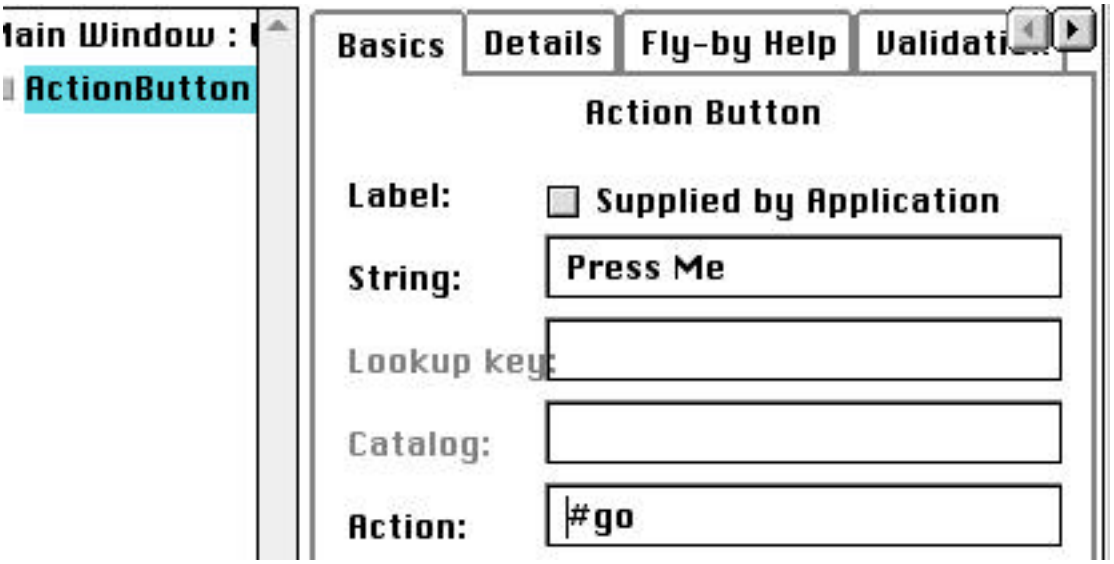
Simple Button Example

In this example we will add a simple button to a window. To add a button to the canvas click on the "ok" button in the palette.

Now click in the canvas where you want the button. You should get a button labeled "Action".



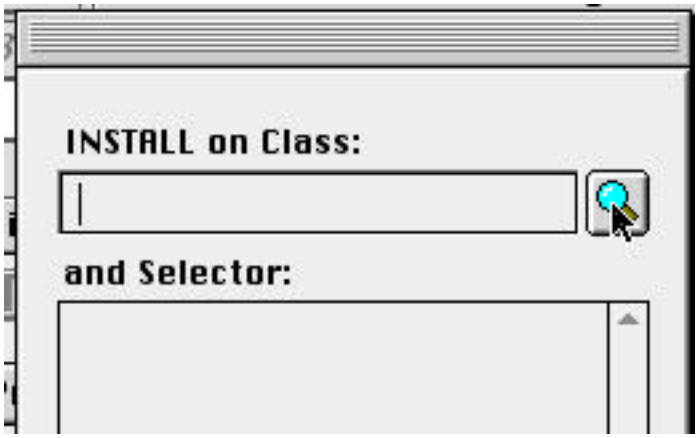
The GUI Painter Tool will also show information about button.



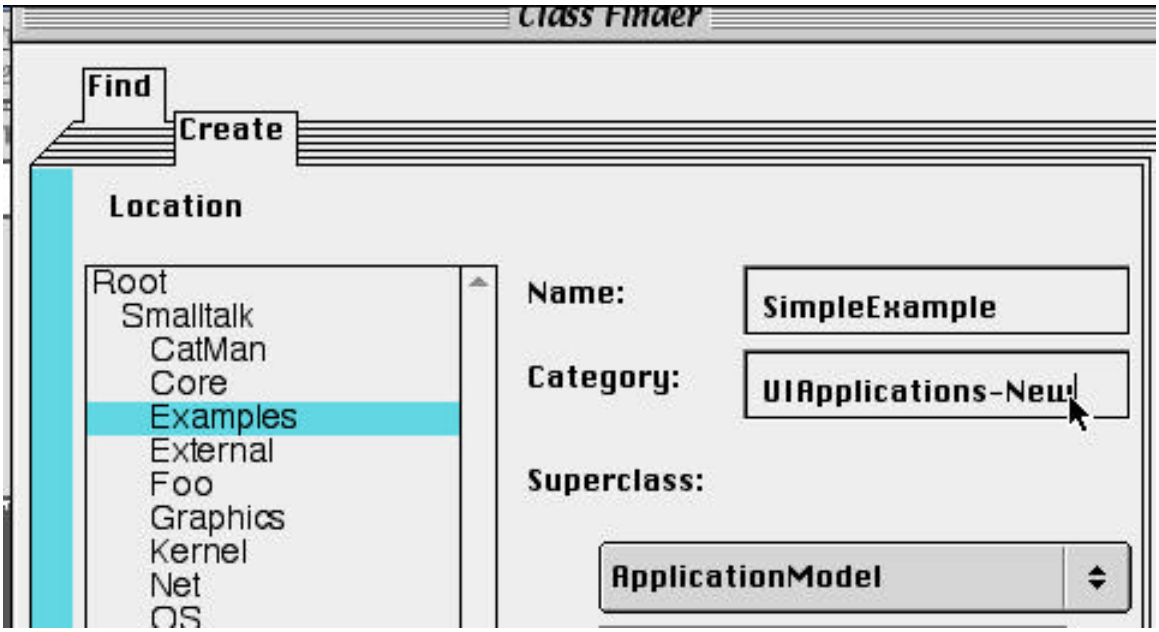
In the GUI Painter Tool, change the String to "Press Me" and the Action: to "go". Actions must be symbols, but if you enter a string the tool will change it to a symbol. Now click on the "Apply" button. The string is the label of the button. The action is the method that will be called when the button is pressed.

Now we need to install the GUI in a class. In the GUI Painter Tool click on the "Install..." button.

You get a dialog:

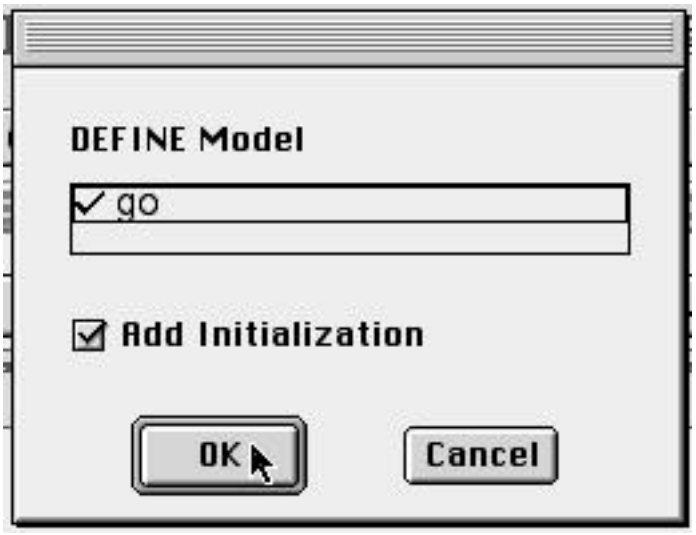


Click on the magnifying glass to open the class finder. We will create a new class. Click on the "Create" tab. Click on the Examples namespace. Then enter the name of the class. I will use "SimpleExample". Now click on the "OK" button to accept. This will exit the class finder and create the class for you. Now click "OK" again to install the window in the class.



We need to make the button do something. In the GUI Painter Tool, with the "Press Me" button selected, click on the "Define…" button.

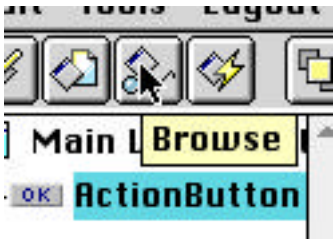
You get the dialog:



Click ok. This defines the method go in the SimpleExample class.

Now to look the methods in SimpleExample. You can use a browser to do this. The GUI Painter Tool provides short cuts to view methods and the class. In the left pane the GUI Painter Tool select the top list item which is labeled "Main Window: Unlabeled". Once that is selected click on the "Browse" icon. You will get a browser on your class.

In the left pane the GUI Painter Tool select the list item for the action button. Once that is selected click on the "Browse" icon. You will get window with all the methods in the class that implement or reference the method go.



The instance method is:

```
SimpleExample>>go

    "This stub method was generated by UIDefiner"

    ^self
```

This does not do anything interesting. Change the method to be:

```
SimpleExample>>go

    "This stub method was generated by UIDefiner"
    Dialog warn: 'Gone'.
    ^self
```

Go back to the GUI Painter Tool. Click on the "Open" button.

You will get a working version of your application. The "Press Me" button will work. Try it. Close the window so we do not get confused with the UI canvas and the working window.

The Class Method: windowSpec

The UIPainter installs a class method called windowSpec. The method returns a string that is used at runtime to create the window. Whenever you make a change to the window in the UIPainter you need to install the change so the UIPainter can update the windowSpec method.

```
SimpleExample class>>windowSpec
    "UIPainter new openOnClass: self andSelector: #windowSpec"

    <resource: #canvas>
    ^#(#{UI.FullSpec}
        #window:
        #(#{UI.WindowSpec}
            #label: 'Unlabeled Canvas'
            #bounds: #(#{Graphics.Rectangle} 464 248 883 368 ) )
        #component:
        #(#{UI.SpecCollection}
            #collection: #(
                #(#{UI.ActionButtonSpec}
                    #layout: #(#{Graphics.Rectangle} 34 33 94 58 )
                    #model: #go
                    #label: 'Press Me'
                    #defaultable: true ) ) ) )
```

Opening the Window

We now have a working application. We saw above how to open the application from the UIPainter. To open the application using Smalltalk code execute the statement in a workspace:

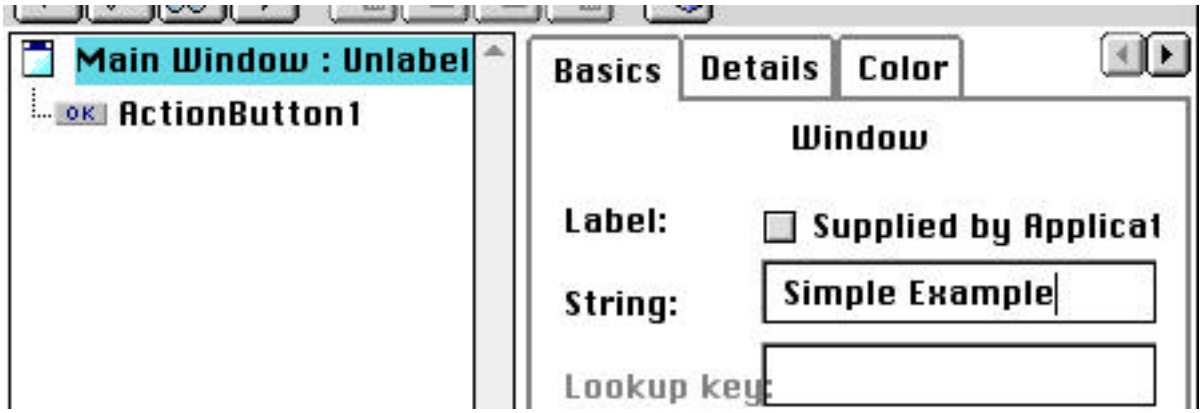
```
SimpleExample open.
```

In other places you may have to specify the full name of the class if the Examples namespace is not imported.

```
Examples.SimpleExample open.
```

Window Label

Let's give the window a better label. In the GUI Painter Tool's left pane select the top item in the list: the Main Window item. Change the text next to the String label to be "Simple Example"

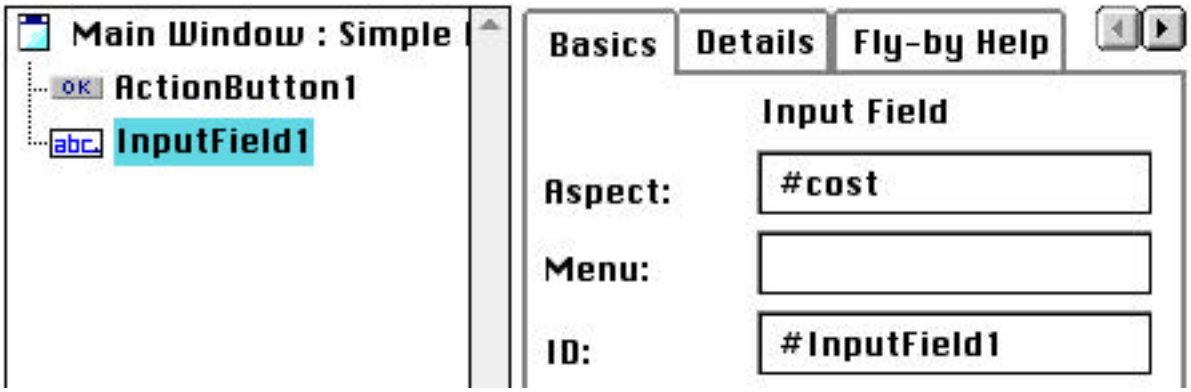


The tittle of the canvas changes. Now click on the "Apply" button on the bottom of the window, and then click on the "Install..." button to update the definition of the window in the class. Then use the "Open" button to get working window.

An Input Field

So far the application does not do much. So let's add an input field. In the palette click on the "Input Field" icon.

Now click in the "Simple Example" canvas where you want the input field.



In the GUI Painter Tool set the aspect to #cost and the type to Number. The aspect is the name of the method in the SimpleExample class the input field will use to get the value to display. The type is what type of input the window will select.

To update the application you need to install the change and then define the method cost. Remember how? First click on the "Apply" button, and then "Install..." button. Once that is over click on the "Define..." button. Once that is done you can use the "Open" button to run the application.

The application window will have an input field. You can type in the input field. The changes are not accepted until you either: hit return, hit tab, or select "accept" in the input field's menu (right click on the input field).

Try entering some text into the numeric input field. The input field will not accept the any text that is not a number.

Connecting a Button to the Input Field

To make the application more interesting lets connect a button to the input field. Now add a button to the canvas. Call the button "Increase" and make its action "#increase". Install the changes and define the increase method. If you don't remember how, just repeat the beginning of this document.

Now we are going to make a few changes in the SimpleExample class to connect the "Increase" button to the input field.

First add the following initialize method to the class:

```
SimpleExample>>initialize
    cost := self cost
```

The cost instance variable and the cost method were created when we installed and defined the input field. Recall that the aspect of the input field is #cost. Now edit the increase method to be:

SimpleExample>>increase

```
"This stub method was generated by UIDefiner"
cost value: (cost value + 1).
^self
```

Once you have made these changes, open the application. Click on the "Increase" button. Magic it works!

Value Holders

Here is how the above program works. We have:

SimipleExample>>cost

```
^cost isNil
    ifTrue:
        [cost := 0 asValue]
    ifFalse:
        [cost]
```

The 0 asValue returns ValueHolder object. In Smalltalk ValueHolder is a model. In Java it would be called an observable. ValueHolders maintain a list of dependents (observers or listeners using Java terminology). When a ValueHolder changes in some way it notifies all its dependents.

When SimpleExample open is called, the "open" method call the new in ApplicationModel which creates a new SimpleExample object and calls its initialize method. Then the open method creates a window and makes the SimpleExample object the window' s model. When the window is created, the input field calls the cost method, getting a reference to the cost ValueHolder. The input field registers as a dependent to the cost ValueHolder. When the increase button is pressed it calls the increase method. This method changes the value of the cost ValueHolder. When this happens, the input field is notified of the change.

If you are a Java programmer note that we did not have to deal with the update notification process.

How do I open the UIPainter on My application?

Now we have a workable GUI application. Close all windows containing your input field and buttons (but don't close the launcher). Now close the canvas and all UIPainter windows.

Now how can we open the UIPainter on our application again? There are two way to do this. One is to go the method

```
SimpleExample class>>windowSpec
    "UIPainter new openOnClass: self andSelector: #windowSpec"

    <resource: #canvas>
    ^#(#{UI.FullSpec}
        #window:
        #(#{UI.WindowSpec}
            #label: 'Simple Example'
            etc.
```

Select the text in the comment (UIPainter new openOnClass: self andSelector: #windowSpec) and then execute it. Do this by right clicking in the window and selecting the "Do it" item in the popup menu. The UIPainter will open on the SimpleExample window.

The other way to open the UIPainter on an existing application is to use the resource finder.

Go to the Launcher and click on the icon:  . This will open the resource finder. You will see a window like:

Find the Examples.SimpleExample class. Select it, the click on the edit button.

Adding Fly-by (Tool tip) Help

There are several different ways to add fly-by help to GUI widgets. We will cover only one way here. In the left pane of the GUI Painter Tool select the widget you wish to add fly-by help to. Now click on the Fly-by Help tab in the right pane. In the input field labeled "Default" type the text for the fly-by help for the widget. Apply the changes and install them in the class. When you now create a new application window, the widget will have fly-by help.

More Information

VisualWorks comes with a number of manuals, guides, examples and tutorials. In your VW directory you should have a doc subdirectory. That directory contains a number of documents in PDF format.

WalkThrough.pdf contains a detailed walkthrough of building an application and using some of the Visual works tools.

The VisualWorks ***GUI Developer's Guide*** contains a lot of information about building GUIs in VisualWorks