CS 535 Object-Oriented Programming & Design
Fall Semester, 2001
Doc 25 Multiple Objects in a GUI

Reference

VisualWorks Application Developer's Guide

Question

Could you tell me how to get two objects to show in one windowSpec?
For example, I have one object from movie class and another from customer class, then how can I show these two objects in one window?

There are several ways of doing this. I will show two different ways.

Solution 1

Three classes will be used in this solution: Foo, Bar and TwoInOneExample. Foo and Bar are the classes whose instances are to be displayed in a GUI on TwoInOneExample. In this solution Foo and Bar do not have window specs and are not subclasses of ApplicationModel. TwoInOneExample has a window spec, which displays Foo and Bar data. First the code, then some comments.

Bar

```
Smalltalk.CS535 defineClass: #Bar
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: 'c d '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

CS535.Bar methodsFor: 'accessing'

```
c
   ^c

c: aString
   c := aString

d
   ^d

d: aString
   d := aString
```

Foo

```
Smalltalk.CS535 defineClass: #Foo
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: 'a b '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'

CS535.Foo methodsFor: 'accessing'

a
   ^a

a: aString
   a := aString

b
   ^b

b: aString
   b := aString
```

TwoInOneExample

```
Smalltalk.CS535 defineClass: #TwoInOneExample
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'myFoo myBar '
   classInstanceVariableNames: ''
   imports: ''
   category: 'UIApplications-New'
```

Instance Methods

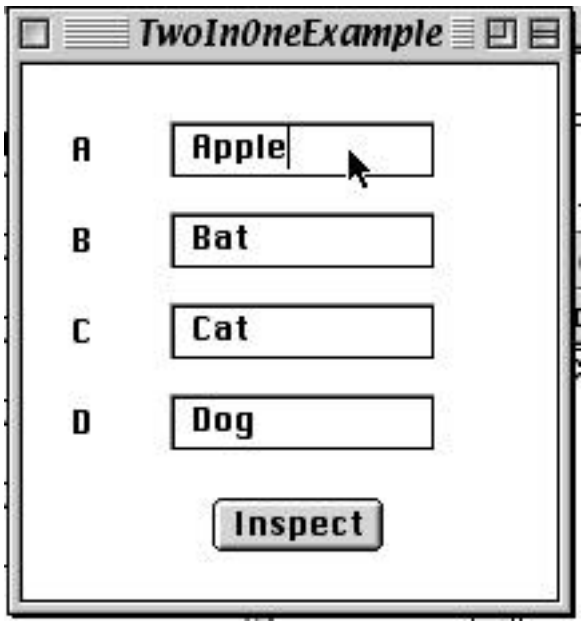
```
myBarValueHolder
   ^myBar asValue

myFooValueHolder
   ^myFoo asValue

initialize
   myBar := Bar new.
   myBar
      c: 'Cat';
      d: 'Dog'.
   myFoo := Foo new.
   myFoo
      a: 'Apple';
      b: 'Bat'.
```

Class Methods

```
windowSpec
    "UIPainter new openOnClass: self andSelector: #windowSpec"
    <resource: #canvas>
    ^#(#{UI.FullSpec}
        #window:
        #(#{UI.WindowSpec}
            #label: 'TwoInOneExample'
            #bounds: #(#{Graphics.Rectangle} 512 384 712 584 ) )
        #component:
        #(#{UI.SpecCollection}
            #collection: #(
                #(#{UI.LabelSpec}
                    #layout: #(#{Core.Point} 16 22 )
                    #name: #Label1
                    #label: 'A' )
                #(#{UI.LabelSpec}
                    #layout: #(#{Core.Point} 16 56 )
                    #name: #Label2
                    #label: 'B' )
                #(#{UI.LabelSpec}
                    #layout: #(#{Core.Point} 16 90 )
                    #name: #Label3
                    #label: 'C' )
                #(#{UI.LabelSpec}
                    #layout: #(#{Core.Point} 16 124 )
                    #name: #Label4
                    #label: 'D' )
                #(#{UI.InputFieldSpec}
                    #layout: #(#{Graphics.Rectangle} 55 21 155 43 )
                    #name: #InputField1
                    #model: #'myFooValueHolder a' )
                #(#{UI.InputFieldSpec}
                    #layout: #(#{Graphics.Rectangle} 55 55 155 77 )
                    #name: #InputField2
                    #model: #'myFooValueHolder b' )
                #(#{UI.InputFieldSpec}
                    #layout: #(#{Graphics.Rectangle} 55 89 155 111 )
                    #name: #InputField3
                    #model: #'myBarValueHolder c' )
                #(#{UI.InputFieldSpec}
                    #layout: #(#{Graphics.Rectangle} 55 123 155 145 )
                    #name: #InputField4
                    #model: #'myBarValueHolder d' )
                #(#{UI.ActionButtonSpec}
                    #layout: #(#{Graphics.Rectangle} 71 162 135 182 )
                    #name: #ActionButton1
                    #model: #inspect
                    #label: 'Inspect'
                    #defaultable: true ) ) ) )
```

About the Example

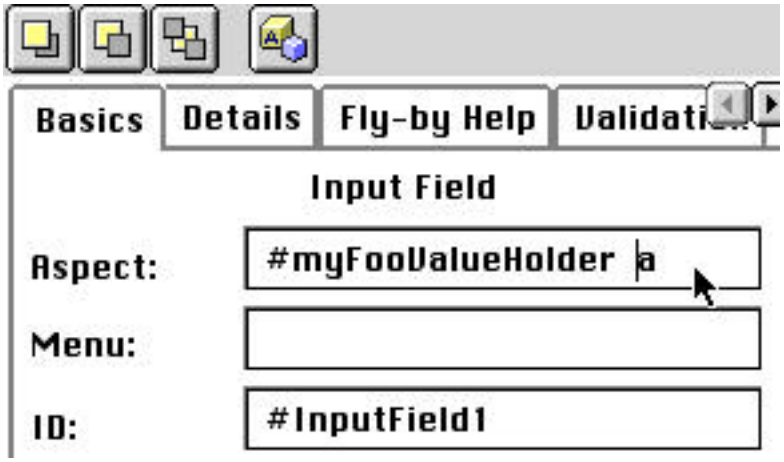
Evaluating "TwoInOneExample open" results in the following window:



Changing the values in the input fields changes the corresponding values in the instance variables in the Foo and Bar objects. The input fields are connected to the instance variables of the Foo and Bar objects. Lets look at how Foo object is connected to the window. First there is the method:

```
myFooValueHolder
  ^myFoo asValue
```

This method returns a value holder on the Foo object. In creating the GUI with the UI Painter we define the aspect of the top input field as:



That is the aspect tells the GUI widget to access the value first call the myFooValueHolder method to get a value holder, get its value (the Foo object) then use the method "a" and "a:" to get and set the values.

Comment 1. The method myFooValueHolder wraps the instance variable in a value holder. As a result if the program changes the value of the Foo object the change will not be reflected in any open windows. If your program needs to change the value of the Foo object then you need to either make the instance variable be a value holder or use adapters.

Comment 2. There are a number of ways to modify this approach. One could add access methods in TwoInOneExample for the instance variable of Foo and Bar. For example:

```
TwoInOneExample>>a
  ^myFoo a

TwoInOneExample>>a: aString
  myFoo a: aString
```

The window spec then could interact with those methods.

Comment 3. As an aid to debugging I often add a "Inspect" button to windows. The aspect of the button is: "#inspect". Do not install an inspect method in your class. Your class inherits an inspect method from Object. This method will open an inspector window on the window's model.

Solution 2 Use Subcanvases

In this solution the Foo and Bar classes have window specs to display their data. The window spec of TwoInOneExample uses the window spec for the Foo and Bar classes. This way the TwoInOneExample does not need to know the details of Foo and Bar. Since Foo and Bar now have window specs, they need to be subclasses of ApplicationModel. They also have to deal with value holders. Here are the modified classes.

Bar

```
Smalltalk.CS535 defineClass: #Bar
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'c d '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

CS535.Bar class methodsFor: 'interface specs'

```
windowSpec
   "UIPainter new openOnClass: self andSelector: #windowSpec"

   <resource: #canvas>
   ^#(#{UI.FullSpec}
      #window:
      #(#{UI.WindowSpec}
         #label: 'Bar'
         #bounds: #(#{Graphics.Rectangle} 512 384 712 584 ) )
      #component:
      #(#{UI.SpecCollection}
         #collection: #(
            #(#{UI.LabelSpec}
               #layout: #(#{Core.Point} 14 12 )
               #name: #Label3
               #label: 'C' )
            #(#{UI.LabelSpec}
               #layout: #(#{Core.Point} 14 50 )
               #name: #Label4
               #label: 'D' )
            #(#{UI.InputFieldSpec}
               #layout: #(#{Graphics.Rectangle} 59 16 159 38 )
               #name: #InputField3
               #model: #cValueHolder )
            #(#{UI.InputFieldSpec}
               #layout: #(#{Graphics.Rectangle} 57 51 157 73 )
               #name: #InputField4
               #model: #dValueHolder ) ) ) )
```

CS535.Bar methodsFor: 'accessing'

```
c
   ^c

c: aString
   c := aString

cValueHolder
   ^c asValue

d
   ^d

d: aString
   d := aString

dValueHolder
   ^d asValue
```

Foo

```
Smalltalk.CS535 defineClass: #Foo
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'a b '
   classInstanceVariableNames: ''
   imports: ''
   category: 'CS535'
```

CS535.Foo class methodsFor: 'interface specs'

```
windowSpec
   "UIPainter new openOnClass: self andSelector: #windowSpec"

   <resource: #canvas>
   ^#(#{UI.FullSpec}
      #window:
      #(#{UI.WindowSpec}
         #label: 'Foo'
         #bounds: #(#{Graphics.Rectangle} 512 384 712 584 ) )
      #component:
      #(#{UI.SpecCollection}
         #collection: #(
            #(#{UI.LabelSpec}
               #layout: #(#{Core.Point} 16 22 )
               #name: #Label1
               #label: 'A' )
            #(#{UI.LabelSpec}
               #layout: #(#{Core.Point} 16 56 )
               #name: #Label2
               #label: 'B' )
            #(#{UI.InputFieldSpec}
               #layout: #(#{Graphics.Rectangle} 55 21 155 43 )
               #name: #InputField1
               #model: #aValueHolder )
            #(#{UI.InputFieldSpec}
               #layout: #(#{Graphics.Rectangle} 55 55 155 77 )
               #name: #InputField2
               #model: #bValueHolder ) ) ) )
```

CS535.Foo methodsFor: 'accessing'

```
a
   ^a

a: aString
   a := aString

aValueHolder
   ^a asValue

b
   ^b

b: aString
   b := aString

bValueHolder
   ^b asValue
```

TwoInOneExample

```
Smalltalk.CS535 defineClass: #TwoInOneExample
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'myFoo myBar '
   classInstanceVariableNames: ''
   imports: ''
   category: 'UIApplications-New'
```

CS535.TwoInOneExample class methodsFor: 'interface specs'

```
usingSubCanvas
    "UIPainter new openOnClass: self andSelector: #usingSubCanvas"

    <resource: #canvas>
    ^#(#{UI.FullSpec}
        #window:
        #(#{UI.WindowSpec}
            #label: 'TwoInOneExample'
            #bounds: #(#{Graphics.Rectangle} 512 384 712 584 ) )
        #component:
        #(#{UI.SpecCollection}
            #collection: #(
                #(#{UI.SubCanvasSpec}
                    #layout: #(#{Graphics.Rectangle} 0 0 205 102 )
                    #name: #Subcanvas1
                    #majorKey: #{CS535.Foo}
                    #minorKey: #windowSpec
                    #clientKey: #foo )
                #(#{UI.SubCanvasSpec}
                    #layout: #(#{Graphics.Rectangle} -1 101 203 201 )
                    #name: #Subcanvas2
                    #majorKey: #{CS535.Bar}
                    #minorKey: #windowSpec
                    #clientKey: #bar ) ) ) )
```

CS535.TwoInOneExample methodsFor: 'initialize'

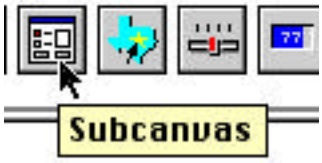
```
initialize
   myBar := Bar new
      c: 'Cat';
      d: 'Dog';
      yourself.
   myFoo := Foo new
      a: 'Apple';
      b: 'Bat';
      yourself
```

CS535.TwoInOneExample methodsFor: 'accessing'

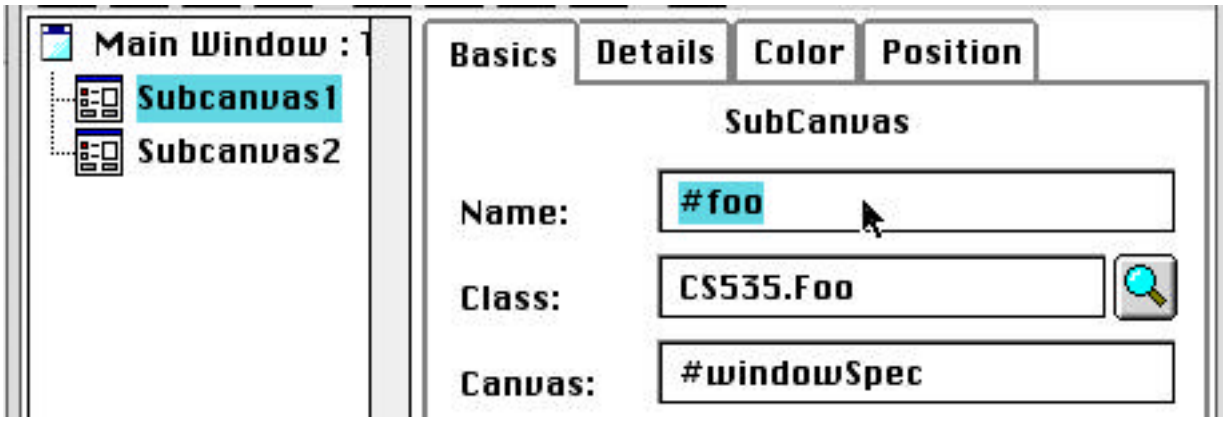
```
bar
   ^myBar

foo
   ^myFoo
```

Creating the Example

There is nothing new in the Foo and Bar classes. What is new is setting up the subcanvas in the TwoInOneExample Class. First we add a subcanvas widget.



Once one has added a subcanvas widget it has to be configured.



There are three important settings: Name, Class, and Canvas.

The Name is the name of the method to call in the TwoInOneExample class that returns the Foo object. Note that we do not return a value holder on a Foo object, but the Foo object itself.

The Class is the class of the object, in this case Foo.

The canvas is the name of the canvas in the Foo class we wish to display in the subcanvas of TwoInOneExample.

Note in this example I have use the name usingSubCanvas for the window spec in TwoInOneExample class.