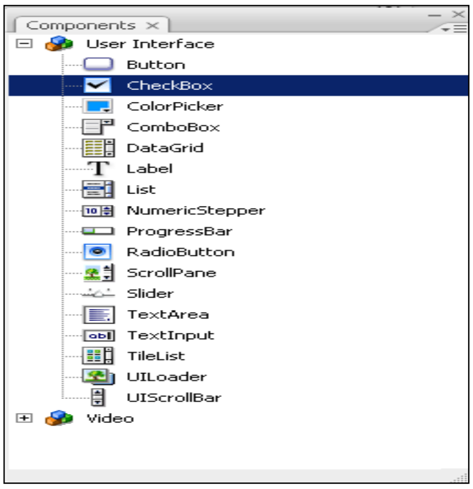
Design for Interaction – Lab 2: ActionScript and GUI

Last week you used buttons to navigate between interfaces, static text boxes to identify the separate interfaces and input text boxes to allow users to enter information. This week you will move on to use other graphical user interface components to allow a user to make more sophisticated choices. It's recommended that you save each of the tasks that you do for later reference. You will also need to type a few lines of Flash's scripting language ActionScript 3 to get some of the interactions to work.

Open a new Flash file in the normal fashion, and ensure it is set to use ActionScript 3.

User Interface Components

Flash provides a range of user interface components which have functionality built into them already. However certain parameters need to be set. Make sure the **Window > Components** is open so that you can see what components are available, and expand the User Interface list. The easiest way of using the components is to drag them from this window onto the stage.



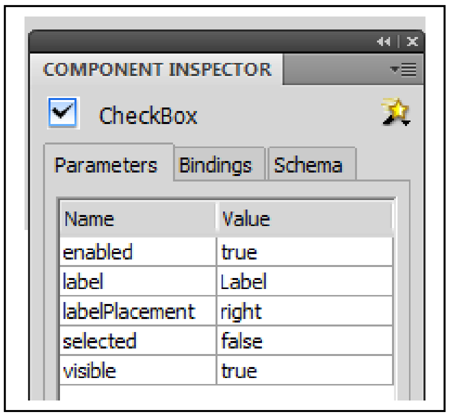
Some of the examples below use a *trace* statement that is printed to an output window during testing when a button is pressed. This is for test purposes only to show how information from the user interface components is obtained – the output window does not appear for real applications and you would need to use the information obtained in a different way.

Button

You have created and used your own buttons and it is perfectly reasonable to continue to do this. However you can use the one provided here – although it is not as fancy as the ones which can be used from ***Windows > Common Libraries > Buttons*** or which you can create yourself. You create the component button by dragging an instance onto the screen. You can edit the label by using the opening the (**Window > Component Inspector**).

Checkbox

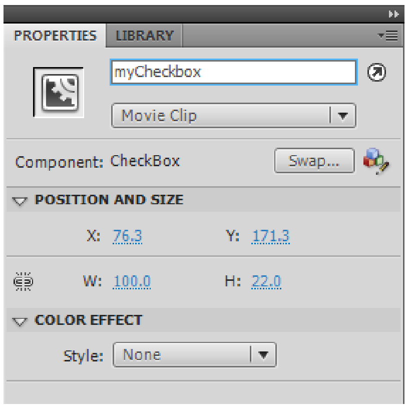
Start a new movie, and drag a check box onto the stage. You can set parameters for your Checkbox by opening the Component Inspector Window (**Window > Component Inspector**). You can change various parameters for the check box by clicking on the values to either reveal an editable field or a drop down menu, as shown below.



Name	Value
enabled	true
label	Label
labelPlacement	right
selected	false
visible	true

- **Label -** edit to whatever label you want on the checkbox
- **Label Placement** - right or left
- **Selected** - true or false (i.e. checked or not)

The check box must be given an instance name in the field in the properties window which shows <Instance Name> by default. I suggest *myCheckbox*. This enables you to keep track of individual check boxes.



myCheckbox

Movie Clip

Component: CheckBox

POSITION AND SIZE

X: 76.3 Y: 171.3

W: 100.0 H: 22.0

COLOR EFFECT

Style: None

Make an input text field with instance name *outputText*. Enter the following script into the actions – frame window in frame 1.

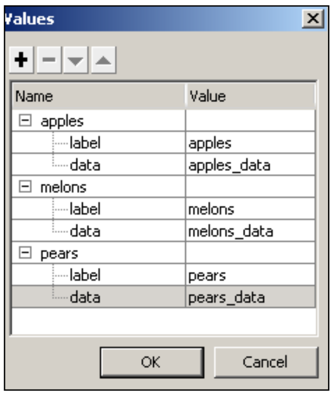
```
myCheckbox.addEventListener(MouseEvent.CLICK, myHandler);

function myHandler(evt:MouseEvent):void {
        outputText.text = String(myCheckbox.selected);
}
```

The first line of the script adds an event listener to the check box instance. The function produces output in a text box showing whether the check box is selected (true) or not (false). Test the movie and see what output text you get for each state of the check box. Based on this test more sophisticated actions can be made.

Combo box (dropdown menu)

Save and close the movie and then start a new one for this next exercise. Drag a combo box onto the stage and give it the instance name *myCombobox*. The parameter you want to edit in the Component Inspector is the dataProvider one – select it and click on the small magnifying glass at the right. From the window that pops up select the + to add a new label and edit the *defaultValue* to read *apples*. Add two more labels *melons* and *pears*. Repeat this for the data parameter and enter *apples_data*, *melons_data* and *pears_data*.



Name	Value
apples	
label	apples
data	apples_data
melons	
label	melons
data	melons_data
pears	
label	pears
data	pears_data

Test the movie – the menu appears and the labels can be selected but other than that it has no functionality. Add two input text fields with instance names *selectedLabel* and *selectedData*, and the following code.

```
myCombobox.addEventListener(Event.CHANGE, my2Handler);

function my2Handler(evt:Event):void {
        selectedLabel.text = myCombobox.selectedItem.label;
        selectedData.text = myCombobox.selectedItem.data;

}
```

Test to check that the appropriate information appears in the text boxes. Add a third input text field called *selectedIndexNo*, and add the following line to your *my2Handler* function after the selectedData.text line.

```
selectedIndexNo.text = String(myCombobox.selectedIndex);
```

On testing the movie again, this text box should show the position of the current selection on the list of options. Note that the top item is number 0, the next is number 1, and so on.

List
The list box is similar in action to the combo box although it looks different (also, it does have the ability to allow multiple selections). Repeat the last exercise and check that you understand the functionality of the component.

Interface Exercise
Save your functioning List box movie and then save it again with a different name so that you can work on it further. Get into the habit of doing this. You will now make a movie where the user selects a fruit from the list and then clicks the button to take them to a new screen showing some information on the fruit.

1. Name the layer you are working in the *interface* layer.
2. Add keyframes in the interface layer in frames 2, 3 and 4
3. Delete the components from frames 2, 3 and 4 but make sure that they remain in frame 1.
4. Add an *actions* layer and type stop(); in the actions for frame 1.
5. Add a *labels* layer and insert keyframes in frames 2, 3 and 4 and label them *apples*, *melons* and *pears*. Label frame 1 *introduction*.
6. "Fruitify" frames 2, 3 and 4 by adding something that identifies the frame with the named fruit in the interface layer.

Edit the ActionScript

```
myList.addEventListener(Event.CHANGE, myHandler);

function myHandler(evt:Event):void {
        var fruit:String = myList.selectedItem.label;
        if(fruit == "apples"){gotoAndStop("apples");}
        if(fruit == "melons"){gotoAndStop("melons");}
        if(fruit == "pears"){gotoAndStop("pears");}
}
```

Save and test your movie. This code tests the label of the selected item in the list and checks to see if it is the same as the text "apples". If it is it causes the movie to jump to the frame labelled *apples*. If the button is clicked does the movie jump to the apples screen?

Test your movie for functionality again – actually you have to test it 3 times, once for each fruit. Why not add a button to each fruit page to return you to the first page?

Radio Button

Start a new movie. Drag 3 radio buttons onto the stage. You will see from the Component Inspector window that they all have the same default Group Name *radioButtonGroup*. This links all these buttons together so that only one can be selected at a time. Various groups can be created in the same frame but for now we will only consider one and this name can be kept as it is. Edit the labels on the buttons to the names of three different fruits. Give the radio buttons instance names *rb1*, *rb2* and *rb3*. Create an input text field with instance name *outputText*. Add the following to the actions window for frame 1.

```
rb1.group.addEventListener(Event.CHANGE,changeHandler);

function changeHandler(evt:Event):void {

        outputText.text = rb1.group.selection.label;
}
```

What more?

If you have difficulty completing the exercises above, please get help with understanding the problem areas.

If you have been successful with the above exercises then you now know the basic functionality to allow you to build an interactive application. Try and build a simple store front end for a retail outlet of your choice, with links to screens showcasing different product types of lines (e.g. a furniture shop could be divided into chairs, beds, sofas, etc.). Use the Flash drawing tools and images from the web to generate suitable visual content for the site.

Note:

As a rule, it is preferable to create your own custom buttons as library symbols rather than using the default Flash components. Not only can you be more visually creative this way, but the components are rather buggy, and prone to causing errors in your movie.