Introduction to Flash III

Nir Peer
CMSC434 TA
Fall 2008

UI COMPONENTS
UI Components – Overview

- Flash provides a collection of standard form components.
  - Available through the Components window (see right).
- Some components on a stage are shown below:

- Adding components may increase the project size even if they are not actually used.
  - You can remove unused components from the Library.
    - Notice also the Component Assets folder.
- Setting a component’s instance name allows it to be referenced by ActionScript code.

Button (1)

- The following code makes the button respond to click events:

```actionscript
btn.addEventListener(MouseEvent.CLICK,
    function(evt : MouseEvent) : void {
        trace("Click!");
    });
```

- This example uses an anonymous function as an event handler.
Button (2)

- Button can also be made into toggles (i.e., stay depressed until released by an additional click).
  - Set the button's `toggle` property to `true`.
- Check the button's `selected` property to determine its state.

- Consider the following stage:

  - We can make the button toggle whether the star is draggable.
    - Note: the star is a MovieClip symbol, so it can be referenced from code.
    - We can make the star draggable by calling: `star.startDrag();`
    - We can end dragging by calling: `star.stopDrag();`
- The next slide shows the code...

```javascript
star.addEventListener(MouseEvent.MOUSE_DOWN, function(evt : MouseEvent) : void {
    // Start dragging if enabled and mouse is clicked
    if (btn.selected) {
        star.startDrag();
    }
});

star.addEventListener(MouseEvent.MOUSE_UP, function(evt : MouseEvent) : void {
    // End dragging when mouse button is lifted
    star.stopDrag();
});
```
CheckBox

- A checkbox is a special toggle button that allows selection.
- To determine whether it is checked, look at its `selected` property.

- Consider the interface on the right.
  - The label of each checked checkbox is added to the textarea.
- The code:
  ```javascript
  // Listen to a global change event
  addEventListener(Event.CHANGE, changeHandler);

  function changeHandler(evt : Event) : void {
    var str : String = "";

    if (chk1.selected) { str += chk1.label + "\n"; }
    if (chk2.selected) { str += chk2.label + "\n"; }
    if (chk3.selected) { str += chk3.label + "\n"; }
    txt.text = str;
  }
  ```

ComboBox (1)

- A combobox presents a list of choices.
- Each of its items is comprised of a label and an associated data object.
- We can populate a combobox using the Parameters tab:

- Or, we can populate it using ActionScript, shown on the next slide.
Let’s have the combobox control which shape is shown.

First, populate the combobox, giving a numeric code to each option:

- We’ll later use it as an index to a shape array.

```javascript
var cbo = new ComboBox();
cbo.addItem({label: "Circle", data: 0});
cbo.addItem({label: "Square", data: 1});
cbo.addItem({label: "Star", data: 2});
```

Now, create an array of shapes and hide all shapes besides the first:

```javascript
var shapes = [circle, square, star];
var indexVisible = 0;
for (var item = MovieClip in shapes) {
    item.visible = false;
}
shapes[indexVisible].visible = true;
```

Finally, each time the combobox selection is changed, hide the current shape and make the selected shape visible.

```javascript
cbo.addEventListener(Event.CHANGE, function(evt) {
    // Hide currently visible shape
    shapes[indexVisible].visible = false;

    // Show selected shape
    indexVisible = cbo.selectedItem.data;
    shapes[indexVisible].visible = true;
});
```
RadioButton

- A radiobutton allows choosing between mutually-exclusive options.
  - For exclusion to work properly, related radiobuttons must belong to the same group.

- Assuming similar code to the one shown before, our event handler now becomes:

```javascript
radio.group.addEventListener(Event.CHANGE,
  function(evt : Event) : void {
    shapes[indexVisible].visible = false;
    indexVisible = Number(radio.group.selection.value);
    shapes[indexVisible].visible = true;
  });
```

NumericStepper

- NumericSteppers allow choosing numbers within a certain range and a specified step size.
- Here:
  - The minimum value is 0
  - The maximum is 100
  - The step size is 25
  - The initial value is 0

- The following script uses a numericstepper to control the star's horizontal position:

```javascript
var initialX : Number = star.x;

ns.addEventListener(Event.CHANGE,
  function(evt : Event) : void {
    star.x = initialX + ns.value;
  });
```
TextInput

- TextInput serves as a text field in forms.
- Some useful properties:
  - `displayAsPassword` allows to hide the actual input characters.
  - `editable` determines whether the field is read-only.
  - `maxChars` restricts the length of input.
  - `restrict` constrains the allowed input characters. Some examples:
    - `ABCabc` – accepts only 'A', 'B', 'C', 'a', 'b', 'c'
    - `A-Z` – accepts only characters in the range 'A' through 'Z'
    - `A-Z0-9` – accepts only characters in the ranges 'A'-'Z' and '0'-'9'
    - `^A-Z` – accepts all characters except the characters 'A' through 'Z'

TextArea

- Similar to TextInput.
- Can handle multiline input.
- Can interpret simple HTML tags (see the `htmlText` property).
A datagrid supports tabular data presentation.

The following code shows how to populate a grid:

```javascript
import fl.controls.dataGridClasses.DataGridColumn;

var colNum : DataGridColumn = new DataGridColumn("Num");
colNum.headerText = "No."

// Add grid columns
grid.addColumn(colNum);
grid.addColumn("Name");
grid.addColumn("Quantity");

// Add data rows
grid.addItem({Num:1, Name:"Jackets", Quantity:3});
grid.addItem({Num:2, Name:"Pants", Quantity:7});
grid.addItem({Num:3, Name:"Shirts", Quantity:24});
```

DataGrids can also be populated using a DataProvider

UILoader

- The UILoader is a container that can display SWF, JPEG, PNG, and GIF files.
- Consider a form containing a UILoader instance named `loader`.
  The following code uses it to display the Google logo.

```javascript
loader.scaleContent = false;

loader.addEventListener(Event.COMPLETE, completeHandler);
function completeHandler (evt : Event) {
    trace("Loaded " + loader.bytesLoaded + " bytes");
}
```

References

- [http://www.adobe.com/devnet/flash/?navID=quickstart](http://www.adobe.com/devnet/flash/?navID=quickstart) and in particular: