Building an Effective User Interface

In this unit, you will learn how to create and manage user interfaces using ActionScript.

Adobe

Objectives

After completing this unit, you should be able to:

- ▶Handle key events on a device
- ▶Set up listeners and event handlers for events
- ▶Use ActionScript to control focus
- ▶Use handlers to track when user input elements gain and lose focus

Handling User Events on a Device

- Events are signals broadcast within the Flash Lite player, notifying that the user has interacted with the device, or that a system event has occurred.
- Much ActionScript programming consists of writing event handlers, which are developer-written functions called by Flash Lite in response to specified events.
- In ActionScript 2.0, a broadcaster/listener model is sometimes used in event handling. The benefit of this approach, where available, is that it allows event handlers to be used for multiple events. Key elements of this model are:
  - Broadcaster – The object on which the event occurs, which also tracks all Listener objects it is to notify.
  - Listener – An object which is notified when an event occurs on another object. The listener is registered with the broadcaster to be notified when an event occurs.
  - Event handler – A developer written function invoked when a specified event occurs.

Handling key events in Flash Lite 2

Flash Lite generates key press events in response to the user pressing keys on their device. A developer writes event handler functions to respond to these key press events, using the `getCode()` method of the `Key` class.

Note: The terms "function" and "method" are virtually interchangeable. A "method" is a function attached to a particular object, or class of objects, such as the `Key` *class.*

Understanding the Key.getCode() method

The `getCode()` method of the `Key` class returns either a constant value - such as `Key.ENTER` - or a `Number` value, depending on what key the user has pressed.

The following table lists constants and values for commonly used device keys:

Select Key	Key.ENTER
Up Navigation Key	Key.UP
Down Navigation Key	Key.DOWN
Left Navigation Key	Key.LEFT
Right Navigation Key	Key.RIGHT
Left Soft Key	ExtendedKey.SOFT1
Right Soft Key	ExtendedKey.SOFT2
0	48
1	49
2	50

Note: The values returned by `Key.getCode()` *may vary somewhat by device. Always test your application in all targeted devices.*

Implementing the broadcaster/listener model in Flash Lite 2

To implement event handling model in Flash, one straightforward approach is to do the following:

1. Create an object from the generic `Object` class to serve as the listener:

```
var keyListener:Object = new Object();
```

2. Create the event handler method as a method of the listener.

```
keyListener.onSpecificEvent = function():Void
{
 //ActionScript code
}
```

Implementing the broadcaster/listener model in Flash Lite 2 (continued)

3. Register the listener to be notified when a specific object occurs on a broadcaster object, in this case the static `Key` class:

```
Key.addListener(keyListener);
```

The process that occurs when an event fires is:

1. A user or system interaction happens with the broadcasting object.
2. The broadcasting object notifies its registered listener(s) that the specific event has occurred.
3. The listener's event handler method is invoked.

Implementing the broadcaster/listener model in Flash Lite 2 (continued)

The code could appear as follows for a soft key press:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function():Void
{
 // code runs if any button is clicked
 trace("A button was clicked!");
}
Key.addListener(keyListener);
```

Responding to key events with conditional logic

A conditional statement can used to control the flow of a program based on the value of an expression, such as a variable or the value returned from a method call. ActionScript 2 supports three forms of conditional statement:

1. `if` / `else` statement
2. `switch` / `case` statement
3. conditional operator (not covered)

Responding to key input using an if / else statement

One or more `if` / `else` blocks may be used to control the program flow, by testing the value returned from the `Key.getCode()` method. For example:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function():Void{
 var keyCode:Object = Key.getCode();
 if (keyCode == Key.ENTER){
  trace ("Selection key was pressed");
 }
 else if (keyCode == Key.LEFT || keyCode == Key.RIGHT){
  trace ("Left or Right key was pressed");
}

  trace ("Some other key was pressed");
}
Key.addListener(keyListener)
```

Responding to key input using a switch / case statement

A `switch` / `case` statement allows testing of multiple possible values for the same expression, such as `Key.getCode()`. For example:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function():Void{
 var keyCode:Object = Key.getCode();
 switch (keyCode){
  case ExtendedKey.SOFT1:
   trace ("SoftKey 1 pressed");
   break; // stop processing here
  case ExtendedKey.SOFT2:
   trace ("SoftKey 2 pressed");
   break; // stop processing here
  default:
   trace ("Some other key pressed");
Key.addListener(keyListener)
```

Communicating with the device

- Flash Lite communicates with its host device and system using the `fscommand2()` global function.

Summarizing fscommand2() settings

- The `fscommand2()` function controls the device, and may retrieve information from the device.
- If a device does not support a given command, the value `-1` is returned when it is called.
- Most commands require one or more additional parameters.

```
status = fscommand2("CommandName", "Param1", "Param2");
```

Note: The return value of `fscommand2()` *need only be captured in a variable, as shown, to conditionally respond to it. It may also be called as a free-standing function.*

Summarizing fscommand2() settings

The following table is a partial list of `fscommand2()` commands. See the documentation for a full command listing.

Partial List of fscommand2() commands

Command Name	Behavior
`GetDevice`	Returns `String` identifying the device type
`GetPlatform`	Returns `String` identifying platform or operating system
`GetVolumeLevel`	Returns `Number` indicating current device volume setting
`SetFocusRectColor`	Sets RGB value for the on-screen focus rectangle
`StartVibrate`	Start device vibrate feature

Setting the Soft Key labels

In Flash Lite 2.x:

- the left soft key is referred to as "SOFT1" or `ExtendedKey.SOFT1`.
- the right soft key is referred to as "SOFT2" or `ExtendedKey.SOFT2`.
- Flash Lite 2.x will support up to 10 additional soft keys, as provided by the device manufacturer (check device documentation).
- The visual labels for the left, right, and other soft keys are set using the `SetSoftKeys` fscommand2 command. For example, to set the left soft key to "Quit" and the right soft key to "Send":

```
fscommand2("SetSoftKeys", "Quit", "Send");
```

Walkthrough 1: Handling Key Presses

In this walkthrough, you will perform the following tasks:

- Capture `Key` press events
- Control view states base on user input

Guiding the User's Experience

Two methods to guide the user's experience with an application:

- ▶Control and respond to the currently selected visual element.
- ▶Control and color the device focus rectangle.

Customizing the focus rectangle

- The focus rectangle is a default yellow highlight that indicates which button or input text box is currently selected.
- Movie clips are also included if their `tabEnabled` property is set to `true`, or if they have event handlers associated with them and their `tabEnabled` property is not set to `false`.

Disabling the focus rectangle

- You can disable the default focus rectangle behavior by setting the global `_focusrect` property to `false`.
- You can disable the focus rectangle for specific `Button` or `MovieClip` objects, by setting their `_focusrect` property to `false`.

```
// prevent selection of the score display
display_mc._focusrect = false;
```

Coloring the focus rectangle

- You can also change the color of the focus rectangle from the default yellow to any other color.
  - To do this you use the `SetFocusRectColor` `fscommand`, which takes RGB (Red, Green, Blue) values, in a range from 0 to 255 for each value, as parameters.
  - For example, the following code changes the color of the focus rectangle to red:

```
fscommand2("SetFocusRectColor", 255, 0, 0);
```

Controlloing focus with the Selection class

- The static `Selection` class lets you set and control the text field in which the insertion point is located (that is, the field that has focus).
- `Selection` indices are zero-based (for example, the first position is 0, the second position is 1, and so on).

For example, to force focus to a specific button:

```
Selection.setFocus(login_btn);
```

Using the onSetFocus event handler

The `onSetFocus` event handler of the `Selection` class is invoked when the input focus changes.

- The `oldFocus` parameter is a reference to the object that lost focus. If there is no previously focused object, `oldFocus` is `undefined`.

The following example demonstrates how you can execute statements when the user of a Flash Lite 2 SWF file moves focus from one button to another.

- Create two buttons, `btn1_btn` and `btn2_btn`, and enter the following ActionScript in Frame 1 of the Timeline:

```
Selection.setFocus(btn1_btn);
trace("initial focus:" + Selection.getFocus());
btn2_btn.onSetFocus = function(oldFocus:Object):Void
{
    trace(oldFocus._name + ":lost focus");
  trace(Selection.getFocus() + ":got focus");
}
```

Referring to code in another timeline

- The main document timeline behaves like a MovieClip.
  - It contains variables, functions, and other code.
  - Within the main document timeline, or the timeline of any MovieClip, all variable, function, and instance names must be unique.
- ActionScript within one timeline can refer to variables, functions, or instance names in another timeline through absolute or relative pathing.

Understanding the path keywords

ActionScript includes three keywords, which may be used anywhere to refer to the paths:

- Current timeline (`this`)
- Main document timeline (`_root`)
- Timeline immediately above the current timeline (`_parent`)

Using this

The keyword `this` refers to the current timeline, wherever the keyword is used. It is often omitted, as any variable, function, or instance name is assumed to be within the current timeline, unless a path keyword is used to indicate a different location.

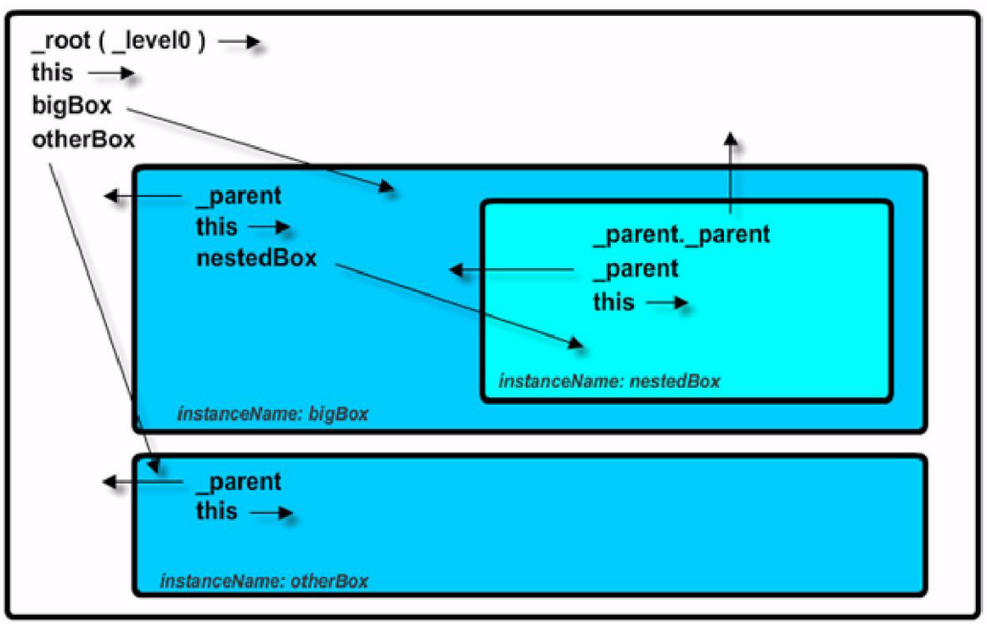

Figure 1: Using the path keywords

Adobe

Using _root

The keyword `_root` is an absolute reference to the main document timeline. From this reference,

- You may refer to and use any variable, function, or instance name on that timeline.
- You may follow the `_root` reference with a series of instance names, to refer to a more deeply nested object.

The use of absolute paths is discouraged, as it leads to tightly coupled code: a change in any name involved in the path will break code located elsewhere within the document.

Absolute Paths

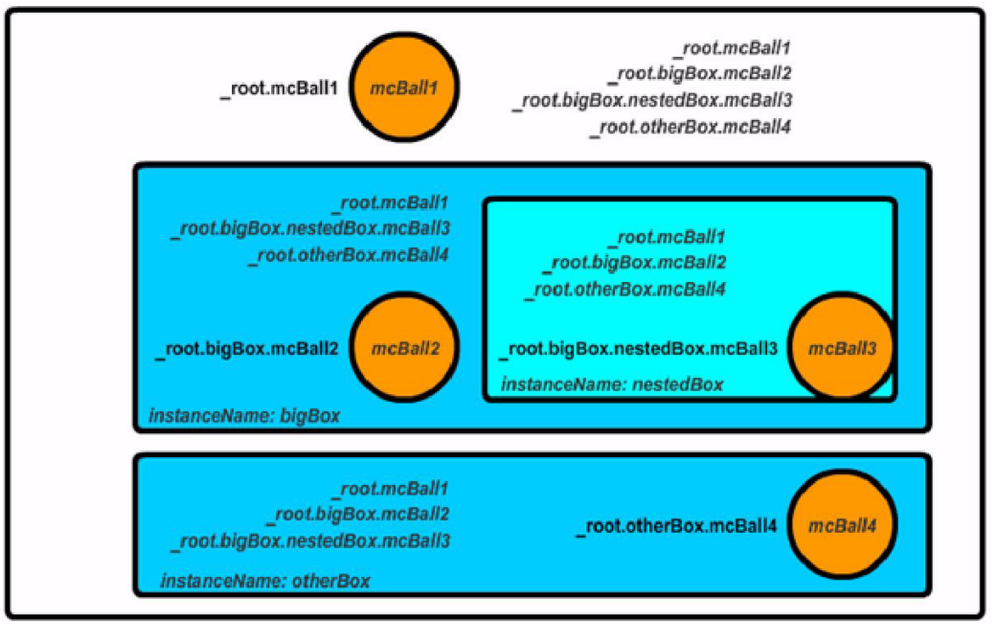

Figure 2: Absolute Paths (Not Recommended)

Adobe

Using _parent

The keyword `_parent` is a relative reference to the timeline immediately above the current timeline.

- Multiple `_parent` references may be connected to refer to timeline more than one above the current timeline.
- From any `_parent` reference, you may refer to and use any variable, function, or instance name of that timeline.

Relative paths

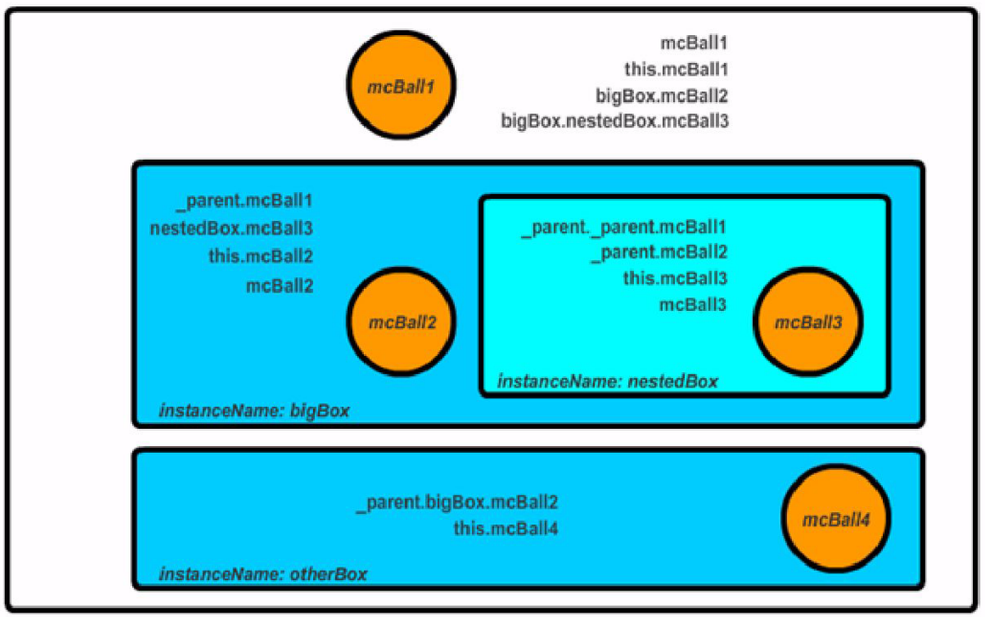

Figure 3: Relative Paths (Recommended)

Adobe

Walkthrough 2: Controlling the Focus Rectangle

In this walkthrough, you will perform the following tasks:

- ▶Capture the user's name
- ▶Customize the focus rectangle
- ▶Turning off the focus rectangle
- ▶Setting focus with ActionScript
- ▶Automatically activate a text field

Summary

- An event is a signal broadcast to notify when the user or system has interacted with an application.
- An event handler is a function - a named block of code - run automatically when a specified event occurs.
- Some (but not all) events can be broadcast to a separate listener object.
- `Key.getCode()` returns a value indicating which key the user has pressed. These values will vary somewhat by device.
- Conditional statements - `if` / `else` and `switch` / `case` - evaluate expressions: variables and the values returned from methods or functions.
- The `fscommand2()` function allows Flash Lite to interact with the device in which it's running, to set soft key labels, cause vibration, etc. Support varies by device.
- The `_focusrect` property allows focus to be disabled for an object.

Summary

- The `Selection` class has methods and events to control focus and determine what object has, or has lost, focus.
- The device focus rectangle can be color-controlled using:

```
fscommand2("SetFocusRectColor", red:Number, green:Number,
  blue:Number);
```