1 The Flash Interface and ActionScript 3.0

Introduction

ActionScript 3.0 is the programming language used in this book, and Flash is the multimedia environment that will be used with ActionScript 3.0. This chapter introduces Flash and examines the essential tools and visual components for building basic multimedia. It also takes a first look at the fundamental structure of an ActionScript 3.0 program.

The chapter tutorial introduces the anatomy of an ActionScript 3.0 program. This tutorial illustrates the creation and execution of a complete Flash application containing simple multimedia objects controlled by an ActionScript 3.0 program. This tutorial is required for learning the essentials of working with both ActionScript 3.0 and Flash in a development environment.

1.1 What Are Flash and ActionScript 3.0?

Flash is a software application that is widely used for creating animations and eye-catching cross-platform web experiences. It is regarded as a powerhouse for multimedia creation due to its ability to seamlessly combine graphics, audio, and video with interactive content in the development of engaging applications such as online games, e-learning tools, and database driven web sites. Flash allows developers to build iPhone and other mobile applications directly in Flash Professional CS5.

ActionScript 3.0 (AS3) is the built-in programming language of Flash. This object-oriented programming (OOP) language performs at a high level with rich functionality.

Flash uses ActionScript 3.0 code to interact with its multimedia elements in the development of multifaceted applications. Several versions of ActionScript are available, with ActionScript 3.0 being the latest generation and the topic of this book.

Flash Player is a web browser plug-in that allows users to view Flash animations and applications distributed across the Internet.

This book focuses on using Flash and ActionScript 3.0 in cooperation to design and program multimedia applications that can be accessed over the Internet and as stand-alone applications that run on the desktop. The ActionScript 3.0 applications created in this textbook will be developed using the Flash CS5 Professional integrated development environment.

It is difficult to characterize Flash because it is used by such a diverse array of professionals. For example, cartoonists are attracted to Flash for its animation capabilities. Animation studios are drawn to the software's ability to construct and organize characters and scenes, build sophisticated animations, and reuse these elements again and again. Figure 1-1 shows a linear animation constructed in Flash and played as a movie with a controller bar in a web browser. For web programmers and game designers, the possibilities for interactive applications are enormous. Figure 1-2 shows applications that use ActionScript 3.0 and Flash, such as an online database XML site and online game of billiards featuring collision detection and realistic physics.

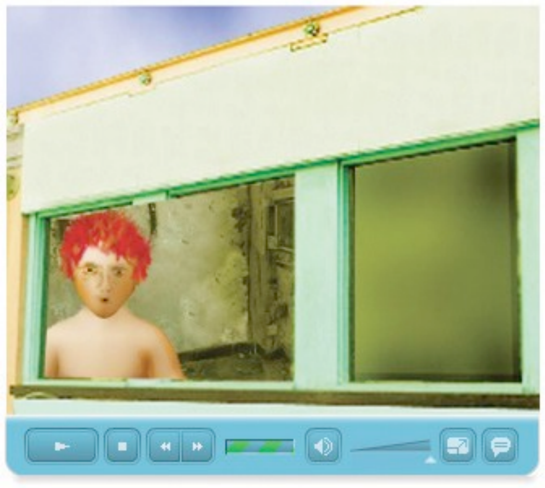


Figure 1-1 An online Flash movie containing a playback controller bar.

Figure 1-2 Examples of online applications constructed using Flash and AS3.

1.2 The Flash Interface: Terminology and Principal Components

In this textbook we will use AS3, along with Flash, to create interactive applications. Thus much of the development work hinges upon an understanding of the Flash visual environment. Designing and implementing applications that use both Flash and AS3 will require considerable switching back and forth between the visual side of Flash and ActionScript programming files.

We begin our tour of Flash by looking at the essential interface elements, which fortunately are few and remarkably intuitive. This section examines the five principal components and basic terminology that are essential to understanding the operational environment. Seasoned Flash users will use more than the five components described in this section to construct sophisticated animations and multimedia. Because this volume is primarily a programming textbook, we will restrict the animation and artwork skills covered here to unadorned basics. This will allow us to commence building visual applications without delay.

The five main interface elements are as follows:

1. Stage
2. Library
3. Timeline
4. Toolbar
5. Properties Inspector

1.2.1 The Stage

Using a theater metaphor, the **Stage** is the scene of action. All graphic content, movement, and interactivity—large or small—will occur on this two-dimensional rectangle. During the creation of a Flash document, the Stage represents the work area where graphics are placed, visual layouts are organized, and movement and interactivity are constructed. The Stage is the only window that end users will actually see.

In Figure 1-3, the Stage is shown as a white rectangle. The Stage uses the common two-dimensional top-left Cartesian coordinate system. This coordinate system is flipped upside down, so that content is contained in the +*x* and +*y* quadrant. The top-left corner, where the *x*-axis and the *y*-axis cross, is designated as the origin of the Stage with an X,Y value of 0,0. Anything to the left of the origin is a negative *x* value and anything above it is a negative *y* value.

The X coordinate of the lower-right corner of the Stage is the width of the Stage. The Y coordinate of the lower-right corner of the Stage is the height of the Stage.

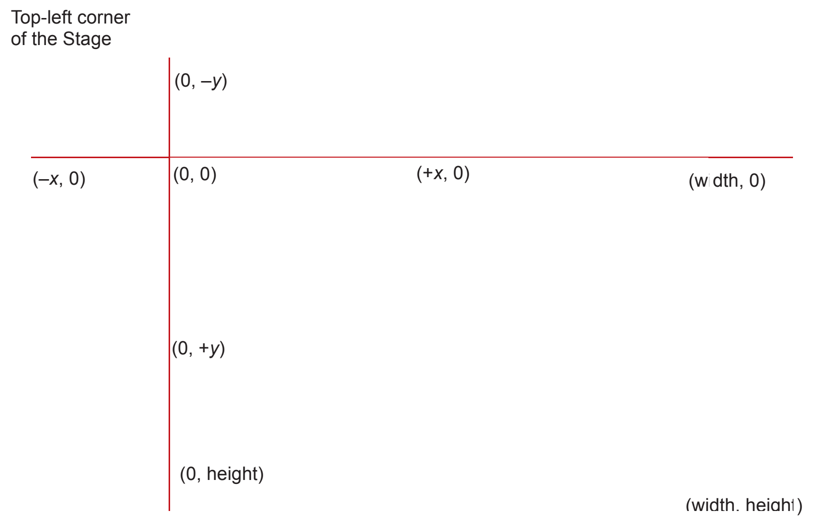


Figure 1-3 The Stage's coordinate system.

Tutorial: Welcome to AS3—"Interactive betty"

Since 1974, it has been a tradition that budding programmers write an introductory program that displays the greeting "Hello World." We will preserve that tradition in this tutorial with a slight modification. We will create a Flash application called "Interactive Betty," shown in Figure 1-41, to interactively display "Hello World" and "Goodbye World." The user will be presented with two buttons. When the user clicks a button, the application will respond by displaying the appropriate greeting or farewell in a dynamic text field.

This tutorial assumes that users have no previous experience with AS3 or computer programming. Our objectives for this tutorial are as follows:

1. Illustrate the process of constructing a simple Flash application from beginning to end in a short number of steps.
2. All Flash applications in this textbook will require two types of files—the Flash file that holds the multimedia elements and the ActionScript 3.0 program file that interacts with them. Both of these files will be constructed in this tutorial.
3. Discuss the idea and importance of the **document class** for a Flash application.
4. Introduce the basic anatomy of an AS3 program.

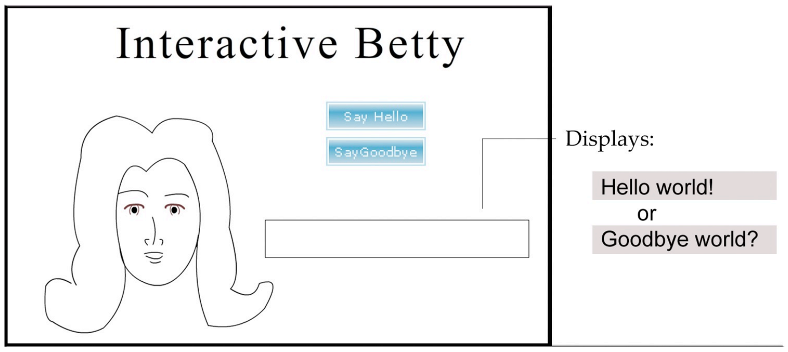


Figure 1-41 The “Interactive Betty” application.

Part I: Creating an Application Program Folder to Hold Project Files

Application programs in Flash typically require many files. For easy navigation, we will place these files in a folder on the desktop.

Step 1: Create a folder on the desktop and name it BettyFolder.

Step 2: Start **Adobe Flash**. Choose **Flash File (AS3)**. Save this new Flash document as bettyApp.fla and locate it in BettyFolder.

Part II: Creating the Visual elements for the Application in Flash

Before we do any programming, we must first create the visual elements. The visual elements of this application are the title for the application, a vector image of Betty, two interactive buttons, and an interactive dynamic text field, which will be used for output. The application title and the image of Betty have no functionality and exist solely for the purpose of making the application visually more pleasing.

Text elements are often created directly on the Stage, will be done in this application. All other elements will be constructed as symbols in the Library.

Task 1: Add the Title “Interactive betty” to the Top of the Stage

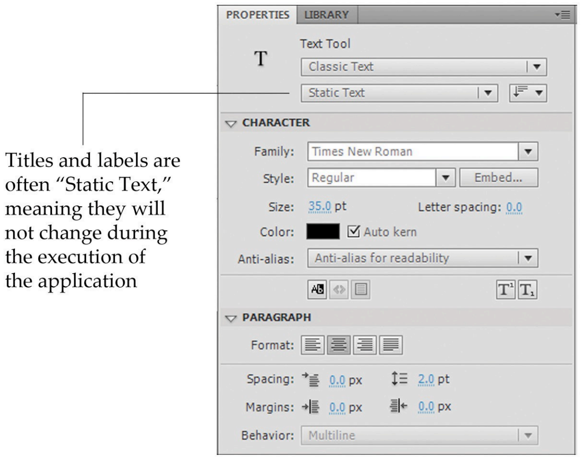
We will start with the title.

Step 1: Choose the **Text tool** from the Toolbar.

Step 2: Before typing any text, we will identify and set the characteristics for this title.

Open the **Properties Inspector** and use Figure 1-42 to guide your selections.

Choose **Static Text** as the text type from the first drop-down menu and set the remaining text properties for **Family**, **Style**, **Size**, **Letter spacing**, and **Color**.



Titles and labels are often “Static Text,” meaning they will not change during the execution of the application

PROPERTIES | LIBRARY

Text Tool

Classic Text

Static Text

CHARACTER

Family: Times New Roman

Style: Regular | Embed...

Size: 35.0 pt Letter spacing: 0.0

Color: ■ ☑ Auto kern

Anti-alias: Anti-alias for readability

PARAGRAPH

Format:

Spacing: 0.0 px 2.0 pt

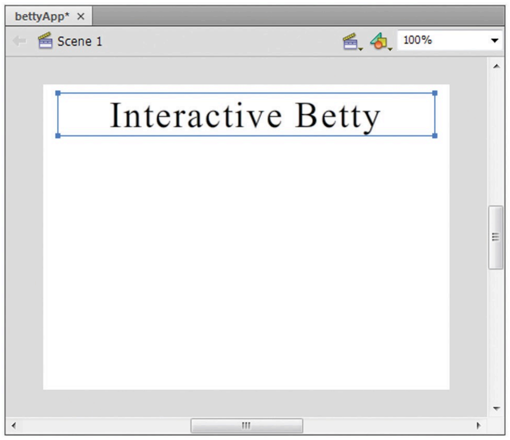
Margins: 0.0 px 0.0 px

Behavior: Multiline

Figure 1-42 The text properties for the title.

Step 3: Click near the top part of the Stage where the title will go and drag the mouse to create a new text box covering almost the full width of the Stage. The text box will appear with a blinking cursor. Type the text **Interactive Betty** as appears in Figure 1-43.

Step 4: Choose **File** > **Save**. Save this file in bettyFolder.



Interactive Betty

Figure 1-43 The title added to the Stage using the Text tool.

Task 2: Add a Dynamic Text Field for Displaying betty's greeting or Farewell

In many applications, it is required that information be displayed in a text field. In this task, an interactive text field of dynamic type will be added to the stage. In addition, we will give this text field a unique identifier name so that the ActionScript 3.0 program to be written in the final steps of this tutorial can access it.

Step 1: Select the **Text tool** from the Toolbar. Open the **Properties Inspector** and choose **Dynamic Text** from the topmost drop-down menu.

Step 2: Drag out the text field on the Stage as shown in Figure 1-44.

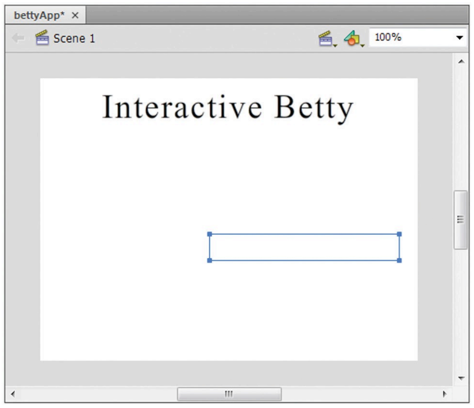
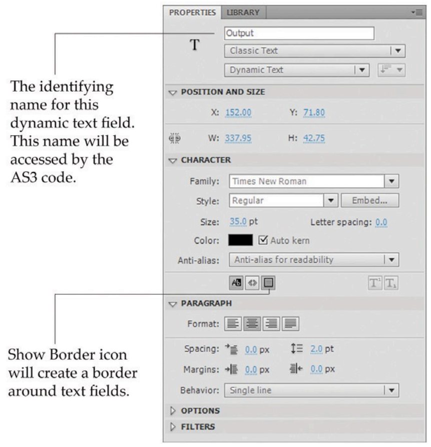


Figure 1-44 A dynamic text field created on the Stage.

Step 3: With the Properties Inspector still open, use Figure 1-45 to specify its name as Output and make the necessary changes to the remaining text properties for **Family**, **Style**, **Size**, **Letter spacing**, and **Color**. Select the Show border icon.


The identifying name for this dynamic text field. This name will be accessed by the AS3 code.
Show Border icon will create a border around text fields.
PROPERTIES
LIBRARY
POSITION AND SIZE
CHARACTER
PARAGRAPH
OPTIONS
FILTERS
Embed...

Output
Classic Text
Dynamic Text
X: 152.00 Y: 71.80
W: 337.95 H: 42.75
Family: Times New Roman
Style: Regular
Size: 35.0 pt Letter spacing: 0.0
Color: ✓ Auto kern
Anti-alias: Anti-alias for readability
Format:
Spacing: 0.0 px 2.0 pt
Margins: 0.0 px 0.0 px
Behavior: Single line

Figure 1-45 The properties for the ~~Output~~ text field on the Stage.

Step 4: Choose **File** > **Save**. Verify this file has been saved in bettyFolder.

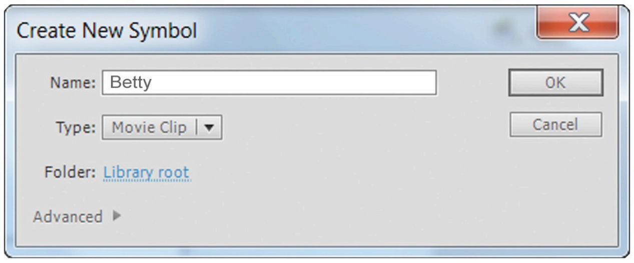
Task 3: The Vector Drawing "betty"

At the moment, the Library is empty. When you are finished with this task, the Library will contain the vector image of Betty as shown in Figure 1-47.

Step 1: Click the **New Symbol icon** in the Library panel.

Step 2: Enter the name Betty.

Step 3: Choose **Movie Clip** from the drop-down menu.(see Figure 1-46) Click **OK**.


Create New Symbol

Name: Betty

Type: Movie Clip

Folder: Library root

Figure 1-46 Dialog box for the Betty symbol.

Step 4: Choose the Paint tools and color of choice to create a vector image of Betty, using Figure 1-47 as a guide.

Step 5: Once the drawing is complete, return to Scene 1.

Step 6: Choose **File** > **Save**. Verify this file has been saved in bettyFolder.

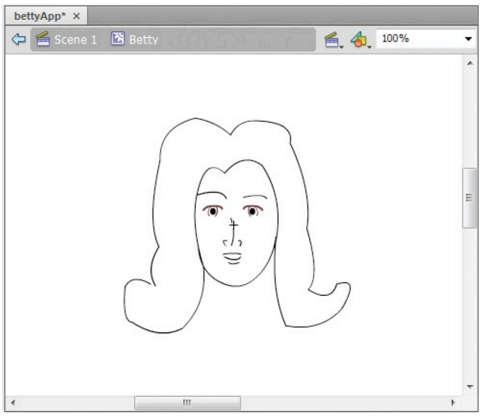


Figure 1-47 Vector drawing of Betty.

Task 4: The Interactive buttons

Buttons are important as interface elements and will be used often in the case studies in future chapters. Adobe Flash provides a collection of predesigned buttons. Instead of creating a button from start to finish, it will be easier to make alterations to an existing one.

Step 1: Choose **Window** > **Common Libraries** > **Buttons**.

Step 2: Locate and select the button named bar BettyApp blue from the Buttons.fla library as in Figure 1-48. Drag this button into the bettyApp library.

Step 3: Within the bettyApp library, rename this button SayHelloBtn. To do so, double-click its name. Drag over a second bar capped blue button and rename this second button SayGoodbyeBtn. Close the **Buttons.fla** library when you are done.

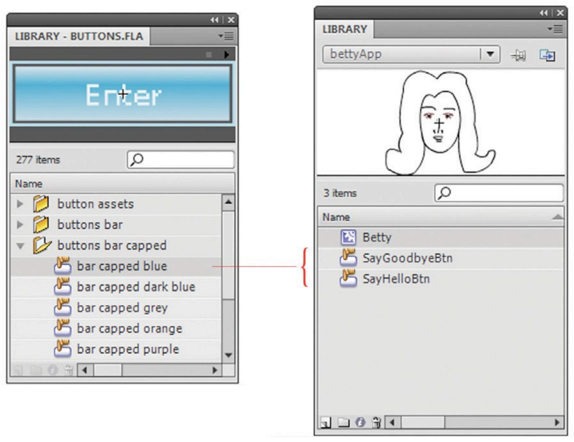


Figure 1-48 The library buttons containing ready-made buttons.

In these next steps we will alter the existing text of the SayHelloBtn button by replacing it with "Say Hello."

Step 4: Select the SayHelloBtn button from the bettyApp library. Double-click it to open it.

When the button appears on the Stage notice, as shown in Figure 1-49, that the Timeline contains four frames corresponding to four possible states: **Up**, **Down**, **Over**, and **Hit**. Drag the Timeline playhead over each state to experience the different look of the button on each frame. The Up state appears when the user's mouse is not on the button; this is the normal state of the button. The Over state shows the appearance of the button when the user rolls the mouse over it. The Down state occurs when the user clicks the mouse while it is over the button. Finally, the Hit frame is a hotspot, the active area of the button that can trigger an action.

Step 5: Select the text layer in the button timeline and use the **Text tool** to replace its current contents with the text **Say Hello**.

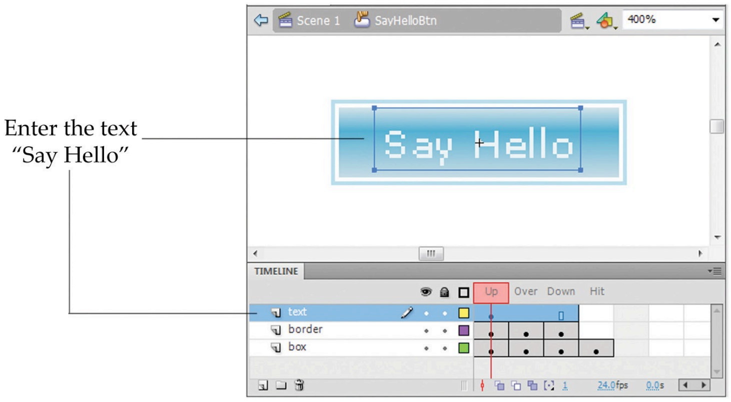


Figure 1-49 The button timeline.

Step 6: Return to Scene 1 and select the SayGoodbyeBtn button from the bettyApp library. Change its text to read **Say Goodbye**.

Step 7: Choose **File** > **Save**. Verify this file has been saved in bettyFolder.

Part III: Completing the Stage Design for use with ActionScript 3.0

In this part of the tutorial, we will complete the visual composition for bettyApp.fla. At this point, the text elements have been placed on the stage and the dynamic text field has been named so the ActionScript 3.0 code can access it. Now it is time to complete the Stage design by placing the button instances on stage and give them names so that our ActionScript 3.0 code can work with them. Betty will also be added to the Stage for visual effect.

Step 1: Verify that Scene 1 is active. With the Library panel open, place an instance of each button on the Stage. Also place an instance of Betty on the Stage. Use Figure 1-50 as a guide for positioning these elements on the Stage.

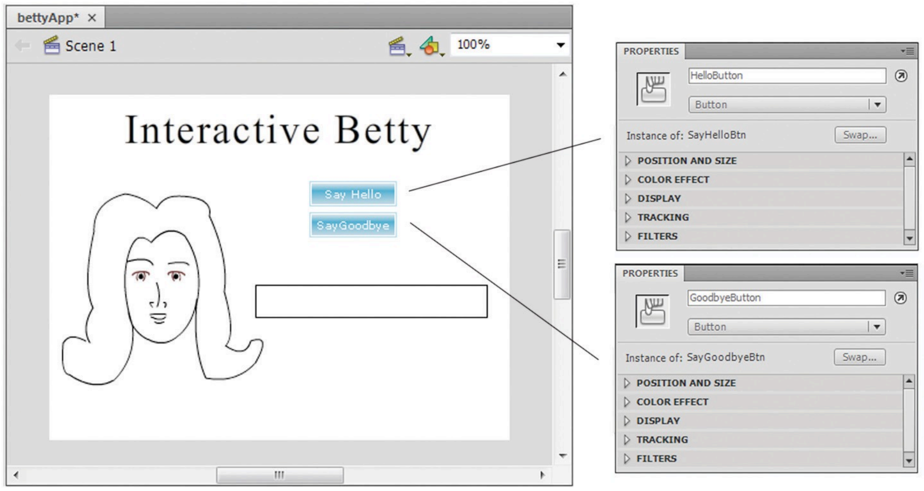


Figure 1-50 Final Stage design with two buttons.

Step 2: With the **Properties Inspector** panel open, select the first button instance and name it HelloButton as shown in Figure 1-50. Select the second button instance and name it GoodbyeButton.

TheActionScript 3.0 program will reference buttons by their instance names.

Step 3: Choose **File** > **Save**.

Part IV: Specifying the Document Class

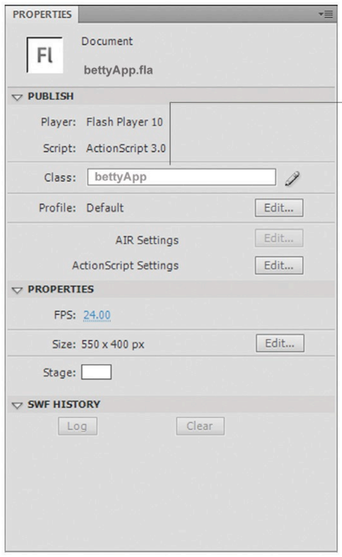
All applications created in this textbook will use a document class. The document class is the main ActionScript 3.0 program file that holds the ActionScript 3.0 code that executes as soon as the application starts. This file is linked to the Flash document and is the engine that drives the application.

As opposed to Flash documents, which have a **.fla** extension, ActionScript 3.0 program files have an **.as** extension. To specify the document class for an application, we simply provide the name of the ActionScript 3.0 program file without the .as extension.

Step 1: With the Properties Inspector open, use the **Selection tool** to click on a blank part of the Stage.

Step 2: Using Figure 1-51 as a guide, enter bettyApp for the **Class** property. This sets the document class to the file named bettyApp.as. Notice that the extension is not included in this name.

The Flash file bettyApp.fla is now linked to the document class bettyApp. The ActionScript 3.0 program file bettyApp.as will now be programmed.


PROPERTIES

Document

bettyApp.fla

PUBLISH

Player: Flash Player 10

Script: ActionScript 3.0

Class: bettyApp

Profile: Default

AIR Settings

ActionScript Settings

PROPERTIES

FPS: 24.00

Size: 550 x 400 px

Stage:

SWF HISTORY

Figure 1-51 Document properties for BettyApp.

Part V: The Anatomy of an ActionScript 3.0 Program

Now we are ready to write the ActionScript code for bettyApp.as. All the applications written in this textbook will be composed of at least two files: the Flash document file and one or more ActionScript 3.0 program files. The complete application for Interactive Betty will consist of the following two files:

- bettyApp.fla
- bettyApp.as

Before we start, a few comments are in order:

1. To keep things simple, we will use a single class file for the first applications in this textbook. The document class will represent this class file.
2. The ActionScript 3.0 program file must be able to locate the Flash document file it will be controlling. For convenience, we will always place these files in the same directory.
3. The code in this part of the tutorial may look rather cryptic. We ask you to simply accept it for now, as some details will not be explained until later in the book.

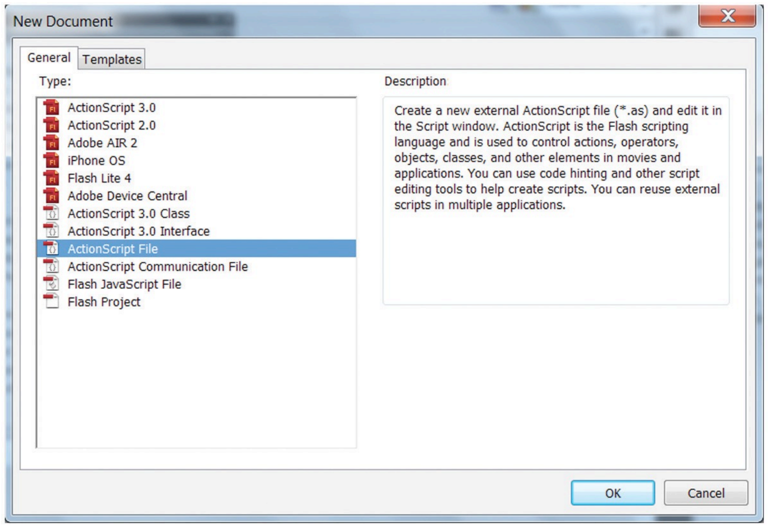
Task 1: Creating an AS (ActionScript) File

Step 1: Choose **File > New**.

Step 2: Choose **ActionScript File** from the **New Document** dialog box, as shown in Figure 1-52.

Step 3: Choose **File** > **Save**. Save this file in the folder bettyFolder. Enter the name bettyApp.as. Click **OK.**

Verify that both bettyApp.as and bettyApp.fla are located in the same directory, bettyFolder.



Create a new external ActionScript file (*.as) and edit it in the Script window. ActionScript is the Flash scripting language and is used to control actions, operators, objects, classes, and other elements in movies and applications. You can use code hinting and other script editing tools to help create scripts. You can reuse external scripts in multiple applications.

Figure 1-52 Dialog box for the new AS3 document.

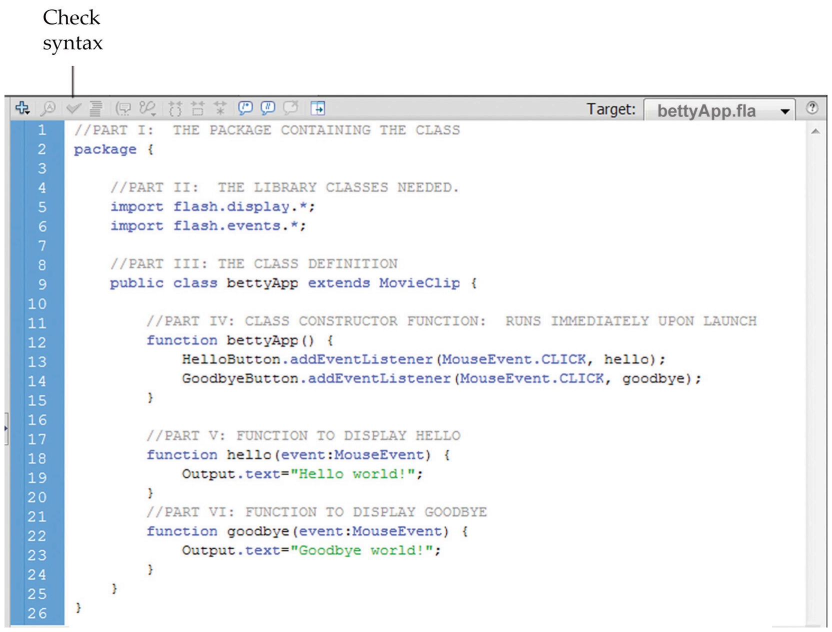
Task 2: Coding and Testing the Application

Step 1: The complete ActionScript 3.0 program file is shown in Figure 1-53. The code must be entered exactly as it appears. Each piece of the program is explained in the anatomy section that follows. Many terms and concepts will be skimmed over here in an attempt provide you will the tools needed to write programs early in the process.

Step 2: Click the **Check syntax icon** to identify any errors. A window will appear indicating your syntax errors, if any. If syntax errors are present, carefully match the entered code with the code provided in Figure 1-53. Each time errors are corrected, check the syntax again until it is error free.

Step 3: Choose **File > Save** to save the bettyApp.as file.

Step 4: Choose **Control** > **Test Movie** to test the application.



```
1  //PART I:  THE PACKAGE CONTAINING THE CLASS
2  package {
3
4      //PART II:  THE LIBRARY CLASSES NEEDED.
5      import flash.display.*;
6      import flash.events.*;
7
8      //PART III: THE CLASS DEFINITION
9      public class bettyApp extends MovieClip {
10
11         //PART IV: CLASS CONSTRUCTOR FUNCTION:  RUNS IMMEDIATELY UPON LAUNCH
12         function bettyApp() {
13             HelloButton.addEventListener(MouseEvent.CLICK, hello);
14             GoodbyeButton.addEventListener(MouseEvent.CLICK, goodbye);
15         }
16
17         //PART V: FUNCTION TO DISPLAY HELLO
18         function hello(event:MouseEvent) {
19             Output.text="Hello world!";
20         }
21         //PART VI: FUNCTION TO DISPLAY GOODBYE
22         function goodbye(event:MouseEvent) {
23             Output.text="Goodbye world!";
24         }
25     }
26 }
```

Figure 1-53 AS3 code for Hello application.

The Anatomy of the bettyApp.as AS3 Program

I: AS3 files are class files. The term **class** will be discussed in the next chapter. A class file always begins with the declaration that it is a **package** containing a class.

```
2    package  {
```

II: Inside the **package** is a list of the library classes that will be imported and used by the program. The first, flash.display.***,** is required to display visual elements and text. The second library, flash.events.*, is needed for the use of interactive buttons. Buttons require an event that listens for the click of a mouse.

```
5    import  flash.display.*;
6    import  flash.events.*;
```

III: The public class bettyApp extends MovieClip line begins the bettyApp class definition. We are keeping matters very simple by using a single class definition for this tutorial, as well as for all case studies until Chapter 9. This class has been *extended* to work with MovieClips, which means that it will work not just with animations and visual graphics, but also, and more specifically, with our Output dynamic text field.

```
9    public class bettyApp  extends MovieClip  {
```

IV: Function bettyApp() is the first function in the program and is the **class constructor**. Later chapters of this book will take an in-depth look at class constructors. The only thing necessary to know at this stage is that this type of function executes automatically when the application runs.

Note three important details:

- A class constructor function should be given the same name as the document class, which is also the same name as the ActionScript file. For example, the name of the class constructor function is bettyApp, and the names of the ActionScript 3.0 file and document class are both bettyApp.
- On line 13, a listener event is initiated to listen for a mouse click of the HelloButton button. Once this button has been clicked, the function displayHello() is called.
- On line 14, a listener event is initiated to listen for a mouse click of the GoodbyeButton button. Once this button has been clicked, the function displayGoodbye() is called.

```
12   function  bettyApp()  {
13       HelloButton.addEventListener(MouseEvent.CLICK,  displayHello);
14       GoodbyeButton.addEventListener(MouseEvent.CLICK,  displayGoodbye);
15   }
```

V: The displayHello() function is a mouse event function. Once this function is called it places the text "Hello world!" into the Output text box.

- On line 19, Output.text uses the dot notation for identifying the text property of the text field named Output. This line simply places the string "Hello world!" in the text box named Output.
- Line 20 has been purposely left blank. We will be adding a trace instruction on this line in the last task of this tutorial.

```
18  function  displayHello(event:MouseEvent)  {
19      Output.text=”Hello world!”;
20
21  }
```

VI: The displayHello() function is a mouse event function. Once this function is called it places the text “Goodbye world!” into the Output text box.

- On line 23, the event that triggers the function is identified as a MouseEvent.
- Line 25 contains a closed curly bracket required to end the displayGoodbye() function definition.

```
23  function displayGoodbye(event:MouseEvent)  {
24      Output.text=”Goodbye  world!”;
25  }
```

The last two lines of the program contain curly brackets. Curly brackets are used to begin and end function definitions, such as with the two functions displayHello() and displayGoodbye(). On line 26, the closed curly bracket ends the public class bettyApp. The last curly bracket of the file closes the package.

```
26      }
27  }
```

Part VI: The trace Statement

Adding a trace statement to AS3 code is invaluable as a debugging tool during the development of ActionScript programs. The trace instruction, which is featured throughout this textbook, is used to display messages, as well as to evaluate and display expressions, in Flash’s Output window. This statement will not affect the finished movie, as it is used only in testing, and the Output window will open only during testing.

In this task we will add a trace statement to display the message “HELLO BETTY” in Flash’s Output window.

Step 1: Verify that you are working in the bettyApp.as document, which is the ActionScript code document. Select the blank line 20 and enter the following instruction:

```
trace(“HELLO  BETTY”);
```

This line will activate the Output window during execution and then display the string “HELLO BETTY” to this window.

Step 2: Execute the application. Click the Say Hello button and verify that two greetings are displayed—one in the Output text field on the Stage and the other in Flash's Output window.

Review Questions

1. Describe the main components of the Flash environment.
2. What is the statement to display a string in a text box?
3. What is a stop action?
4. What do the extensions .as and .fla stand for?
5. What is a listener event?

Exercises

1. Create an interactive movie that contains two buttons and one output text box. When clicked, the first button should display "Goodnight" in the text box. The second interactive button should display "Good morning."

2. Create an interactive movie that contains blue, purple, and salmon buttons from the "buttons circle flat"—Buttons.fla library. Each button will display the text "What color am I?" Include a static text box that displays the text "BLUE, PURPLE, or SALMON?" Modify the code from Tutorial 4 so that each color of button that is clicked displays the name of that color in a dynamic text box. Example: If you click the purple button, the text "PURPLE" appears in the output box.